US Government Protection Profile

Personal Electronic Devices

For

Medium Robustness Environments

Information
Assurance
Directorate

November 2, 2004
Version 0.13

Protection Profile Title:

1 U.S. Government Protection Profile Personal Electronic Devices for Medium Robustness Environments.

Criteria Version:

2 This Protection Profile (PP) was developed using Version 2.2 of the Common Criteria (CC) and applying the National Information Assurance Partnership (NIAP) interpretations that have been approved by Trust Technology Assessment Program/ Common Criteria Evaluation Standard Scheme (TTAP/CCEVS) Management as of June 30, 2004.

Changes from version 0.3 of the PP

1. Changed from Personal Digital Assistant (PDA) to Personal Electronic Device (PED)
2. Eliminated the appendices concept. The Wireless Internet, Wireless E-Mail, Synchronization, and Wireless Networking appendices were incorporated into the base PED.
3. Added a cellular package that includes cellular voice, short message service, and push-to-talk.
4. Deleted most of the PED management requirements. It is assumed that the device will be configured by the administrator and provided to the user. The user will not have access to any of the administrative functions. If any changes to the configuration are needed, the user will have to return the device to the administrator.
5. Auditing was added.

Table of Contents

List of Tables and Figures

1 INTRODUCTION TO THE PROTECTION PROFILE

3 This section contains overview information necessary to allow a Protection Profile (PP) to be registered through a Protection Profile Registry. The “Identification” section provides the labelling and descriptive information necessary to identify, catalo gue, register, and cross-reference a PP. The “Overview” section summarizes the profile in narrative form and provides sufficient information for a potential user to determine whether the PP is of interest. The overview can also be used as a stand-alone abstract for PP catalogues and registers. The “Conventions” section provides the notation, formatting, and conventions used in this protection profile. The “Glossary of Terms” section gives a basic definition of terms, which are specific to this PP. The “Document Organization” section briefly explains how this document is organized

1.1 PP Identification

4 Title: US Government Protection Profile Personal Electronic Devices for Medium Robustness Environment

5 Sponsor: National Security Agency (NSA)

6 CC Version: Common Criteria (CC) Version 2.2, and applicable interpretations.

7 Registration: <to be provided upon registration>

8 PP Version: Version 0.11

9 Keywords: Medium Robustness Environments, Personal Digital Assistant, PDA, Personal Electronic Device, PED

1.2 Overview of the Protection Profile

1.3 Conventions

10 Except for replacing United Kingdom spelling with American spelling, the notation, formatting, and conventions used in this PP are consistent with version 2.2 of the CC. Selected presentation choices are discussed here to aid the PP reader.

11 The CC allows several operations to be performed on functional requirements; *refinement*, *selection*, *assignment*, and *iteration* are defined in paragraph 2.1.4 of Part 2 of the CC. Each of these operations is used in this PP.

12 The **refinement** operation is used to add detail to a requirement, and thus further restricts a requirement. Refinement of security requirements is denoted by **bold text**.

13 The **selection** operation is used to select one or more options provided by the CC in stating a requirement. Selections that have been made by the PP authors are denoted by *italicized text*, selections to be filled in by the Security Target (ST) author appear in square brackets with an indication that a selection is to be made, [selection:], and are not italicized.

14 The **assignment** operation is used to assign a specific value to an unspecified parameter, such as the length of a password. Assignments that have been made by the PP authors are denoted by showing the value in square brackets, [Assignment_value], assignments to be filled in by the ST author appear in square brackets with an indication that an assignment is to be made [assignment:].

15 The **iteration** operation is used when a component is repeated with varying operations. Iteration is denoted by showing the iteration number in parenthesis following the component identifier, (iteration_number).

16 As this PP was sponsored, in part by National Security Agency (NSA), National Information Assurance Partnership (NIAP) interpretations are used and are presented with the NIAP interpretation number as part of the requirement identifier (e.g., **FAU_GEN.1-NIAP-0407** for Audit data generation).

17 The CC paradigm also allows protection profile and security target authors to create their own requirements. Such requirements are termed 'explicit requirements' and are permitted if the CC does not offer suitable requirements to meet the authors' needs. **Explicit requirements** must be identified and are required to use the CC class/family/component model in articulating the requirements. In this PP, explicit requirements will be indicated with the "(EXP)" following the component name.

18 Application Notes are provided to help the developer, either to clarify the intent of a requirement, identify implementation choices, or to define "pass-fail" criteria for a requirement. For those components where Application Notes are appropriate, the Application Notes will follow the requirement component.

1.4 Glossary of Terms

19 See Section 8 for the Glossary.

1.5 Document Organization

20 Section 1, Introduction to the Protection Profile, provides the document management and overview information necessary to identify the PP.

21 Section 2, Target of Evaluation (TOE) Description, defines the TOE and establishes the context of the TOE by referencing generalized security functions.

22 Section 3, Security Environment, describes the expected environment in which the TOE is to be used. This section defines the set of threats that are relevant to the

secure operation of the TOE, organizational security policies with which the TOE must comply, and secure usage assumptions applicable to this analysis.

23 Section 4, Security Objectives, defines the set of security objectives to be satisfied by the TOE and by the TOE operating environment.

24 Section 5, IT Security Requirements, specifies the security functional and assurance requirements that must be satisfied by the TOE and the IT environment.

25 Section 6, Rationale, provides rationale to demonstrate that the security objectives satisfy the threats and policies. This section also explains how the set of requirements are complete relative to the security objectives and presents a set of arguments that address dependency analysis and Strength of Function (SOF) and use of the explicit requirement.

26 Section 7, References, provides background material for further investigation by users of the PP.

27 Section 8, Glossary, provides a listing of definitions of terms.

28 Section 9, Acronyms, provides a listing of acronyms used throughout the document.

29 Section 10, Robustness Environment Characterization, contains a discussion characterizing the level of robustness TOEs compliant with the PP can achieve. The PPRB created a discussion that provides a definition of factors for TOE environments as well as an explanation of how a given level of robustness is categorized.

30 Section 11, Explanatory Material for Explicit Assurance Requirements, provides objectives and application notes for the explicit ADV requirements contained in this PP.

31 Section 12, Refinements, identifies the refinements that were made to CC requirements where text is deleted from a requirement.

2 TOE DESCRIPTION

2.1 Product Type

32 This Protection Profile is written to support the development of a Personal Electronic Device (PED). There are however, an almost limitless number of features that could be added to a device of this type. As a result, PED described in this PP has a core set of features and an optional package of cellular communications. A vendor claiming compliance must include the base functional package and if the vendor's PED has cellular capabilities they must meet the cellular package. The cellular package cannot be selected by itself. The cellular package includes the short-message-service, voice communications, and push-to-talk (walkie-talkie).

33 A PED is a small, handheld mobile device that provides users with computing and storage and retrieval capabilities. The devices covered by this PP are single-user devices with a graphical user interface (GUI). The PED is used primarily as a personal organizer but network connectivity (wired and wireless) is included in the base unit. The network connectivity is provided through connection between the PED and a trusted network.

34 Wireless network connectivity will be between the PED and a trusted access point. The access point and the PED device negotiate keys for mutual authentication and secure communication before permitting any transfer of data. A new key is generated between the wireless access point and the PED at the end of a time interval specified by the administrator.

35 The user may browse the Internet or access e-mail through the network connection. In addition, the user can synchronize email messages, calendar appointments or address book contacts on the PED with a desktop application. Removable storage such as flash memory or USB memory sticks may be provided to assist in the transfer of data between computer systems or as a means to backup user data stored on the device.

36 The e-mail capability allows users to send and receive email. The security features make it possible for users to sign and encrypt email messages they create, as well as verify signatures and decrypt email messages they receive from others. It is still possible to send and receive plaintext, unsigned email messages as well.

37 The Internet browser will have the capability to access both secure and non-secure web sites.

38 Most PEDs have the following as standard features:

- Calendar/ Datebook: The calendar application has the ability to store and retrieve information such as appointments, schedules and meetings.

- Address Book: The address book application stores contact information so that the authorized user may easily access it.

- Notes: A user may store notes by typing them in, or as in the case with some PEDs, use a stylus to write them onto the electronically sensitive screen.

- Keyboard: The keyboard is a "thumb-typing" QWERTY keyboard, designed to allow users to input and update quickly and easily into the Calendar or Notebook.

- Storage: The PED has storage capability for all of the local applications. This storage can be encrypted to protect information on the PED. Removable storage may be provided to allow the transfer of data between the PED and other computer systems.

- Microphone/Speaker

- Network Connectivity (Wired or Wireless)

- Synchronization

- E-Mail

- Web Browsing

- External Interfaces, such as a USB port.

- Other applications.

2.2 TOE Definition

39 The TOE boundary includes the PED device itself (hardware), the operating system, and associated applications. It does not include any of the remote systems the PED may connect to during use.

2.3 General TOE Functionality

40 The PED is a single-user medium robustness device. The PED shall provide the following security services in its evaluated TOE configuration:

41 *Identification and Authentication* – The user must provide I&A data prior to accessing any application on the device. The TOE provides the capability to support password authentication. The TOE enforces several restrictions on authentication data including, password length, strength, and character set. The TOE also provides a lockout capability, which locks the PED after the user has entered an authorized administrator determined number of invalid attempts.

42 *Self-Protection* – The TOE provides domain separation and non-bypassability protection. All software executes in its own domain, and cannot affect the TSF or other programs. All other information on the PED is treated as data and is not considered executable. The device itself must be protected from physical tampering by methods similar to tamper resistant seals.

43 *Administrative Role* – The TOE provides for one administrative role that configures the security policy for the PED as well as manages all TSF data. It is assumed that the device will be configured by the administrator and provided to the user. The user will not have access to any of the administrative functions. If any changes to the configuration are needed, the user will have to return the device to the administrator. The Administrative role will be able to invoke self-tests.

44 *Trusted Path* – The TOE is required to provide a Trusted Path. A Trusted Path refers to the encrypted connection used to authenticate an external human user with the TOE. The TOE must provide a trusted path between the user and the TOE to ensure the user is sending authentication data to the TOE and not a malicious entity.

45 *Encryption* – Cryptographic algorithms and key management functions that meet published standards are required in PED PP-complaint products. Section 5.1.1 “Cryptographic Support” defines the minimum set of cryptographic attributes required by the TOE. The TOE’s cryptographic module(s) must be FIPS PUB 140-2 validated and must meet, at a minimum, the security requirements of “Security Level 1”. The ST author may implement the cryptographic module(s) in hardware, software, or a combination of both. The TOE must generate and distribute symmetric keys. The ST author is provided several implementation selections for key generation and may distribute keys manually, electronically, or both. The TOE must perform data encryption/decryption using the Triple Data Encryption Algorithm (TDEA) algorithm with a minimum key size of 168 bits. Additional requirements for key destruction, generation, verification, random number generation and cryptographic hashing are provided in section 5.1.1.

2.4 TOE Operation Environment

46 This PED device is meant to store Sensitive data. Users will carry this device in many trusted and untrusted environments. There are requirements on the PED that will allow it to be taken into a Sensitive Compartmented Information Facility (SCIF). In SCIF Mode, all import and export communication capabilities and data gathering capabilities are disabled (eg. microphone, camera, LAN connectivity).

2.5 Cellular Communications Package

47 Cellular communication is the focus of the package. A base set of requirements have been established for a PED. Additional functionality in the form of cellular communications can be added to the base PED by incorporating the following requirements in a Security Target (ST). In order to claim compliance with the PED PP with Cellular Communication all the requirements in both the base PP and this

package must be satisfied. Individual features cannot be cut out of the package and included with the base PED PP. The cellular communications package includes secure voice, secure push-to-talk (walkie-talkie), and secure short message service (SMS).

2.5.1 Secure Voice

48 Secure Cellular Voice allows users to communicate by voice to other secure telephones. In general, voice communications are accessed wirelessly via a cellular network. We assume the network is an untrusted IT entity and do not make assumptions about the security of the TOE based on authentication and encryption algorithms for the network. All encryption algorithms and keys for secure voice communications between end users will be stored on the PED device itself.

49 The PED with Voice provides the following additional security services in it's evaluated configuration:

- Encryption – Cryptographic algorithms and operations that meet published standards are required in products compliant with this package. Section 5.1.1 of the PED PP defines the base set of cryptographic requirements for the PED device. These requirements include asymmetric key generation, asymmetric key validation and packaging, cryptographic signature service, availability of digital signature operations, key agreement, and cryptographic signature

2.5.2 Secure Push-to-Talk

50 Secure push to talk over cellular networks utilizes the 'always on' characteristics of the network and enables the functional equivalent to "walkie-talkies" using a PED. This added feature can only be accomplished between two (or more) TOE's, but the default push to talk will be an unencrypted conversation.

51 With this service, a single button on a PED enables brief and immediate voice exchanges without having to place an actual phone call. Unlike short-range radio walkie-talkies, push to talk over cellular networks is not dependent upon distance. Users can be in close proximity or across the country. A push to talk over cellular session can be one user to one user, or one user to many users.

52 Push to talk over cellular makes use of half duplex transmission. Data are transmitted on a single, bi-directional channel. So whether a session is one to one or one to many, there can only be one person speaking at a time. Push to talk over cellular can be considered more convenient and economical than regular cellular calls, because the channel is only kept open for the duration of the "spurts" of conversation.

53 The PED with this capability must provide the following additional security services in its evaluated TOE configuration:

- *Secure data transfer* – Confidentiality and integrity are provided through encryption of the voice transmission between the end users.

54 All encryption algorithms and keys for secure push to talk communications between end users will be stored on the PED device itself.

55 Each device has a unique identifying number for push to talk over cellular. It works by creating a link between devices by using either the phone number of the device or a unique identifying number for the device. This number can be stored by peers and accessed in the same way that cell phone numbers are stored in a phonebook application. Storing several numbers together under a group name can also form talk groups.

2.5.3 Short Message Service

56 The SMS capability for a PED allows users to send and receive text messages using the PED device.

57 A compliant TOE will allow a subscriber to receive SMS text messages from other subscribers. The SMS capability allows a store-and-forward system for short messages. SMS data can be transmitted simultaneously with voice, data or fax calls.

58 SMS is very similar to paging; however, the receiving mobile device does not need to be active and in range for the SMS messages to be sent. They will be held in transit until the designated recipient is available. When the subscriber accesses the receiving device, delivery is attempted at that time. The device is paged and if it responds, the message is delivered and a verification receipt can be sent back to the sending mobile device.

59 The PED with SMS capability shall provide the following security service in its evaluated TOE configuration:

- Secure data transfer – Confidentiality of transmitted data is provided through encryption of the SMS message content. This protects the communication that goes over the air between the sending and receiving PED.

3 SECURITY ENVIRONMENT

60 A medium robustness TOE is considered sufficient protection for environments where the likelihood of an attempted compromise is medium. This implies that the motivation of the threat agents will be average in environments that are suitable for TOEs of medium robustness. Note that this also implies that the resources and expertise of the threat agents really are not factors that need to be considered, because highly sophisticated threat agents will not be motivated to use great expertise or extensive resources in an environment where medium robustness is suitable.

61 The medium motivation of the threat agents can be reflected in a variety of ways. One possibility is that the value of the data processed or protected by the TOE will be only medium, thus providing little motivation of even a totally unauthorized entity to attempt to compromise the data. Another possibility, (where higher value data is processed or protected by the TOE) is that the procuring organization will provide environmental controls (that is, controls that the TOE itself does not enforce) in order to ensure that threat agents that have generally high motivation levels (because of the value of the data) cannot logically or physically access the TOE (e.g., all users are "vetted" to help ensure their trustworthiness, and connectivity to the TOE is restricted).

62 The remainder of this section addresses the following:

a) Threats to TOE assets or to the TOE environment which must be countered;

b) Organizational Security Policies that compliant TOEs must enforce;

c) Assumptions about the security aspects of a compliant TOE environment.

3.1 Threats

3.1.1 Threat Agent Characterization

63 In addition to helping define the robustness appropriate for a given environment, the threat agent is a key component of the formal threat statements in the PP. Threat agents are typically characterized by a number of factors such as *expertise*, *available resources*, and *motivation*. Because each robustness level is associated with a variety of environments, there are corresponding varieties of specific threat agents (that is, the threat agents will have different combinations of motivation, expertise, and available resources) that are valid for a given level of robustness. The following discussion explores the impact of each of the threat agent factors on the ability of the TOE to protect itself (that is, the robustness required of the TOE).

64 The *motivation* of the threat agent seems to be the primary factor of the three characteristics of threat agents outlined above. Given the same expertise and set of resources, an attacker with low motivation may not be as likely to attempt to compromise the TOE. For example, an entity with no authorization to low value data none-the-less has low motivation to compromise the data; thus a basic robustness TOE should offer sufficient protection. Likewise, the fully authorized user with access to highly valued data similarly has low motivation to attempt to compromise the data, thus again a basic robustness TOE should be sufficient.

65 Unlike the motivation factor, however, the same can't be said for *expertise*. A threat agent with low motivation and low expertise is just as unlikely to attempt to compromise a TOE as an attacker with low motivation and high expertise; this is because the attacker with high expertise does not have the motivation to compromise the TOE even though they may have the expertise to do so. The same argument can be made for *resources* as well.

66 Therefore, when assessing the robustness needed for a TOE, the motivation of threat agents should be considered a "high water mark". That is, *the robustness of the TOE should increase as the motivation of the threat agents increases.*

67 Having said that, the relationship between expertise and resources is somewhat more complicated. In general, if resources include factors other than just raw processing power (money, for example), then expertise should be considered to be at the same "level" (low, medium, high, for example) as the resources because money can be used to purchase expertise. Expertise in some ways is different, because expertise in and of itself does not automatically procure resources. However, it may be plausible that someone with high expertise can procure the requisite amount of resources by virtue of that expertise (for example, hacking into a bank to obtain money in order to obtain other resources).

68 It may not make sense to distinguish between these two factors; in general, it appears that the only effect these may have is to lower the robustness requirements. For instance, suppose an organization determines that, because of the value of the resources processed by the TOE and the trustworthiness of the entities that can access the TOE, the motivation of those entities would be "medium". This normally indicates that a medium robustness TOE would be required because the likelihood that those entities would attempt to compromise the TOE to get at those resources is in the "medium" range. However, now suppose the organization determines that the entities (threat agents) that are the least trustworthy have no resources and are unsophisticated. In this case, even though those threat agents have medium motivation, the likelihood that they would be able to mount a successful attack on the TOE would be low, and so a basic robustness TOE may be sufficient to counter that threat.

69 It should be clear from this discussion that there is no "cookbook" or mathematical answer to the question of how to specify exactly the level of motivation, the amount of resources, and the degree of expertise for a threat agent so that the robustness

level of TOEs facing those threat agents can be rigorously determined. However, an organization can look at combinations of these factors and obtain a good understanding of the likelihood of a successful attack being attempted against the TOE. Each organization wishing to procure a TOE must look at the threat factors applicable to their environment; discuss the issues raised in the previous paragraph; consult with appropriate accreditation authorities for input; and document their decision regarding likely threat agents in their environment.

70 The important general points are:

a) The motivation for the threat agent defines the upper bound with respect to the level of robustness required for the TOE.
b) A threat agent's expertise and/or resources that are "lower" than the threat agent's motivation (e.g., a threat agent with high motivation but little expertise and few resources) may lessen the robustness requirements for the TOE (see next point, however).
c) The availability of attacks associated with high expertise and/or high availability of resources (for example, via the Internet or "hacker chat rooms") introduces a problem when trying to define the expertise of, or resources available to, a threat agent.

71 The following threats are addressed by the TOE and should be read in conjunction with the threat rationale, Section 6.1. There are other threats that the TOE does not address (e.g., malicious developer inserting a backdoor into the TOE) and it is up to a site to determine how these types of threats apply to its environment.

Table 1 Medium Robustness Applicable Threats

Threat Name	Threat Definition
T.ADMIN_ERROR	An administrator may incorrectly install or configure the TOE, or install a corrupted TOE resulting in ineffective security mechanisms.
T.AUDIT_COMPROMISE	A malicious user or process may view audit records, cause audit records to be lost or modified, or prevent future audit records from being recorded, thus masking a user's action.

Threat Name	Threat Definition
T.CRYPTO_COMPROMISE	A malicious user or process may cause key, data or executable code associated with the cryptographic functionality to be inappropriately accessed (viewed, modified, or deleted), thus compromising the cryptographic mechanisms and the data protected by those mechanisms.
T.FLAWED_DESIGN	Unintentional or intentional errors in requirements specification or design of the TOE may occur, leading to flaws that may be exploited by a malicious user or program.
T.FLAWED_IMPLEMENTATION	Unintentional or intentional errors in implementation of the TOE design may occur, leading to flaws that may be exploited by a malicious user or program.
T.LOSS_OR_THEFT	Portable devices might be lost or stolen allowing a malicious user to use that device to gain sensitive information
T.MALICIOUS_TSF_COMPROMISE	A malicious user or process may cause TSF data or executable code to be inappropriately accessed (viewed, modified, or deleted).
T.MASQUERADE	A malicious user, process, or external IT entity may masquerade as an authorized entity in order to gain access to data or TOE resources.
T.POOR_TEST	Lack of or insufficient tests to demonstrate that all TOE security functions operate correctly (including in a fielded TOE) may result in incorrect TOE behavior being undiscovered thereby causing potential security vulnerabilities.

Threat Name	Threat Definition
T.REPLAY	A user may gain inappropriate access to the TOE by replaying authentication information, or may cause the TOE to be inappropriately configured by replaying TSF data or security attributes (e.g., captured as transmitted during the course of legitimate use).
T.RESIDUAL_DATA	A user or process may gain unauthorized access to data through reallocation of TOE resources from one user or process to another.
T.RESOURCE_EXHAUSTION	A malicious process or user may block others from system resources via a resource exhaustion denial of service attack.
T.SPOOFING	A malicious user, process, or external IT entity may misrepresent itself as the TOE to obtain identification and authentication data.
T.UNATTENDED_SESSION	A user may gain unauthorized access to an unattended session.
T.UNAUTHORIZED_ACCESS	A process may gain access to user data for which it is not authorized according to the TOE security policy.
T.UNIDENTIFIED_ACTIONS	The administrator may fail to notice potential security violations, thus limiting the administrator's ability to identify and take action against a possible security breach.
T.UNKNOWN_STATE	When the TOE is initially started or restarted after a failure, the security state of the TOE may be unknown.

Table 2 Medium Robustness Threats Not Applicable to the TOE

Threat Name	Threat Definition	Rationale
T.ADMIN_ROGUE	An administrator's intentions may become malicious resulting in user or TSF data being compromised.	This threat is not applicable to this TOE because it is a single user device with one administrative role. Therefore if the administrator does become rogue there will be no one else to stop them.

3.2 Organizational Security Policies

72 An organizational security policy is a set of rules, practices, and procedures imposed by an organization to address its security needs

Table 3 Medium Robustness Applicable Policies

Policy Name	Policy Definition
P.ACCESS_BANNER	The TOE shall display an initial banner describing restrictions of use, legal agreements, or any other appropriate information to which users consent by accessing the TOE.
P.ACCOUNTABILITY	The authorized users of the TOE shall be held accountable for their actions within the TOE.
P.ADMIN_ACCESS	Administrators shall be able to administer the TOE locally through protected communications channels.
P.CRYPTOGRAPHIC_FUNCTIONS	The TOE shall provide cryptographic functions for key management and cryptographic services.

Policy Name	Policy Definition
P.CRYPTOGRAPHY_VALIDATED	Where the TOE requires FIPS-approved security functions, only NIST FIPS validated cryptography (methods and implementations) are acceptable for key management (i.e.; generation, access, distribution, destruction, handling, and storage of keys) and cryptographic services (i.e.; encryption, decryption, signature, hashing, key distribution, and random number generation services).
P.ENCRYPT_STORED_DATA	The TOE shall encrypt all user data that is stored within the TOE using TDEA and a key size of 168 bits.
P.PDA_PKI	DOD class 4, Version 3 X.509 certificates shall be used as appropriate for encryption and to digitally sign transmissions.
P.SCIF_MODE	The TOE must not collect or record any audio or video data or emit electronic communications while in a SCIF.
P.TRANSPORT_PROTECTION	The TOE shall provide encryption and signature services to protect user data while it is being transmitted to and from the TOE.
P.VULNERABILITY_ANALYSIS_TEST	The TOE must undergo appropriate independent vulnerability analysis and penetration testing to demonstrate that the TOE is resistant to an attacker possessing a medium attack potential.

Table 4 Medium Robustness Policies Not Applicable to the TOE

Policy Name	Policy Definition	Rationale
P.CRYPTOGRAPHY	The TOE shall use NIST FIPS validated cryptography as a baseline with additional NSA-approved methods for key management (i.e., generation, access, distribution, destruction, handling, and storage of keys), and for cryptographic operations (i.e., encryption, decryption, signature, hashing, key exchange, and random number generation services).	This policy was replaced with P.CRYPTOGRAPHY _VALIDATED

3.3 Assumptions

73 This section contains assumptions regarding the security environment and the intended usage of the TOE.

Table 5 Medium Robustness Applicable Assumptions

Assumption Name	Assumption Definition
A.ENVIRONMENT_PKI	Within the IT environment there is a Public Key Infrastructure (PKI) infrastructure that provides valid Class 4 Version 3 X.509 certificates which users are trained to use properly.
A.NO_TAMPER	The TOE must show visual evidence of any physical tampering.
A.PHYSICAL	It is assumed that the IT environment provides the TOE with appropriate physical security, commensurate with the value of the IT assets protected by the TOE.
A. SCIF_MODE	It is assumed that the authorized user will put the PED device into SCIF mode before entering the SCIF and will only take it out of SCIF mode upon leaving the SCIF.

Assumption Name	Assumption Definition
A.SINGLE_USER	It is assumed that the administrator of the TOE will configure the TOE with only one user, non-administrative, account.

Table 6 Medium Robustness Assumptions Not Applicable to the TOE

Assumption Name	Assumption Definition	Rationale
A.NO_GENERAL_ PURPOSE	The administrator ensures there are no general-purpose computing or storage repository capabilities (e.g., compilers, editors, or user applications) available on the TOE.	A PED is not a server-type product so there will be user applications and editors residing within the TOE.

4 SECURITY OBJECTIVES

74 This section identifies the security objectives of the TOE and its supporting environment. The security objectives identify the responsibilities of the TOE and its environment in meeting the security needs.

4.1 TOE Security Objectives

Table 7 Medium Robustness Security Objectives

Objective Name	Objective Definition
O.ADMIN_ROLE	The TOE will provide an administrator role to isolate administrative actions.
O.ADMIN_GUIDANCE	The TOE will provide the administrator with the necessary information for secure delivery and management.
O.AUDIT_GENERATION	The TOE will provide the capability to detect and create records of security-relevant events associated with users.
O.AUDIT_PROTECTION	The TOE will provide the capability to protect aud it information.
O.AUDIT_REVIEW	The TOE will provide the capability to selectively view audit information, and alert the administrator of identified potential security violations.
O.CHANGE_MANAGEMENT	The configuration of, and all changes to, the TOE and its development evidence will be analyzed, tracked, and controlled throughout the TOE’s development.
O.CORRECT_TSF_OPERATION	The TOE will provide a capability to test the TSF to ensure the correct operation of the TSF in its operational environment.
O.CRYPTOGRAPHIC_FUNCTIONS	The TOE shall provide cryptographic functions for its own use, including encryption/decryption and digital signature operations.

Objective Name	Objective Definition
O.CRYPTOGRAPHY_VALIDATED	The TOE shall use NIST FIPS 140-2 validated cryptomodules for cryptographic services implementing FIPS-approved security functions and random number generation services used by cryptographic functions.
O.DISPLAY_BANNER	The TOE will display an advisory warning regarding use of the TOE
O.DOCUMENT_KEY_LEAKAGE	The bandwidth of channels that can be used to compromise key material shall be documented.
O.ENCRYPT_STORED_DATA	All user data stored on the TOE will be encrypted using TDEA with a key size of 168 bits.
O.MAINT_MODE	The TOE shall provide a mode from which recovery or initial startup procedures can be performed
O.MANAGE	The TOE will provide all the functions and facilities necessary to support the administrators in their management of the security of the TOE, and restrict these functions and facilities from unauthorized use.
O.MEDIATE	The TOE must protect user data in accordance with its security policy.
O.REPLAY_DETECTION	The TOE will provide a means to detect and reject the replay of authentication data, as well as, TSF data and security attributes.
O.RESIDUAL_INFORMATION	The TOE will ensure that any information contained in a protected resource is not released when the resource is reallocated.
O.RESOURCE_SHARING	The TOE shall provide a mechanism that mitigates attempts to exhaust resources provided by the TOE.

Objective Name	Objective Definition
O.ROBUST_TOE_ACCESS	The TOE will provide mechanisms that control a user's logical access to the TOE and to explicitly deny access to specific users when appropriate
O.SELF_PROTECTION	The TSF will maintain a domain for its own execution that protects itself and its resources from external interference, tampering or unauthorized disclosure.
O.SOUND_DESIGN	The TOE will be designed using sound design principles and techniques. The TOE design, design principles and design techniques will be adequately and accurately documented.
O.SOUND_IMPLEMENTATION	The implementation of the TOE will be an accurate instantiation of its design, and is adequately and accurately documented
O.THOROUGH_FUNCTIONAL_TESTING	The TOE will undergo appropriate security functional testing that demonstrates the TSF satisfies the security functional requirements.
O.TIME_STAMPS	The TOE shall provide reliable time stamps and the capability for the administrator to set the time used for these time stamps.
O.TRUSTED_PATH	The TOE will provide a means to ensure that users are not communicating with some other entity pretending to be the TOE when supplying identification and authentication data.
O.USER_GUIDANCE	The TOE will provide users with the information necessary to correctly use the security mechanisms

Objective Name	Objective Definition
O.VULNERABILITY_ANALYSIS_TEST	The TOE will undergo appropriate independent vulnerability analysis and penetration testing to demonstrate the design and implementation of the TOE does not allow attackers with medium attack potential to violate the TOE's security policies

4.2 Environment Security Objectives

Table 8 Medium Robustness Environmental Security Objectives

Environmental Objective Name	Environmental Objective Definition
OE.ENVIRONMENT_PKI	Those responsible for the TOE will ensure the TOE uses a PKI infrastructure that provides valid class 4, Version 3 X.509 certificates. Additionally, users will be properly trained in the operations and procedures of PKI.
OE.NO_TAMPER	Those responsible for the TOE will apply tamper resistant seals to the TOE to allow visual detection of physical tampering
OE.PHYSICAL	Physical security will be provided within the domain for the value of the IT assets protected by the operating system and the value of the stored, processed, and transmitted information.
OE.SCIF_MODE	The TOE will have a means of placing the PED securely into SCIF mode and to remain in this mode while in the SCIF.
OE.SINGLE_USER	It is assumed that the administrator of the TOE will configure the TOE with only one user, non-administrative, account.

5 IT SECURITY REQUIREMENTS

5.1 TOE Security Functional Requirements

75 This section defines the functional requirements for the TOE. Functional requirements in this PP were drawn directly from Part 2 of the CC, or were based on Part 2 of the CC. These requirements are relevant to supporting the secure operation of the TOE.

Table 9 Security Functional Requirements

Functional Components (from CC Part 2)	
FAU_ARP.1	Security alarms
FAU_ARP_ACK_(EXP).1	Security alarm acknowledgement
FAU_GEN.1-NIAP-0407	Audit data generation
FAU_GEN.2-NIAP-0410	User identity association
FAU_SAA.1-NIAP-0407	Potential violation analysis
FAU_SAR.1	Audit review
FAU_SAR.2	Restricted audit review
FAU_SAR.3	Selectable audit review
FAU_SEL.1-NIAP-0407	Selective audit
FAU_STG.1-NIAP-0429	Protected audit trail storage
FAU_STG.3	Action in case of possible audit data loss
FAU_STG.NIAP-0414	Site-configurable prevention of audit data loss
FCS_BCM_(EXP).1	Baseline cryptographic module
FCS_CKM.1(1)	Cryptographic key generation (symmetric)
FCS_CKM.1(2)	Cryptographic key generation (asymmetric)
FCS_CKM.2	Cryptographic key distribution

Functional Components (from CC Part 2)	
FCS_CKM.4	Cryptographic key destruction
FCS_CKM_(EXP).1	Cryptographic key validation and packaging
FCS_CKM_(EXP).2	Cryptographic key handling and storage
FCS_COA_(EXP).1	Cryptographic operations availability
FCS_COP.1(1)	Cryptographic operation (Data encryption/decryption)
FCS_COP.1(2)	Cryptographic operation (Cryptographic signature)
FCS_COP.1(3)	Cryptographic operation (Cryptographic hashing)
FCS_COP.1(4)	Cryptographic operation (Cryptographic key agreement)
FCS_COP_(EXP).1	Random number generation
FDP_ACC.2(1)	Complete access control (Stored data policy)
FDP_ACC.2(2)	Complete access control (Application access Control Policy)
FDP_ACF.1(1)	Security attributes based access control
FDP_ACF.1(2)	Security attributes based access control
FDP_IFC.1	Subset information flow control (External Flow Policy)
FDP_IFC.2(1)	Complete information flow control (Application Separation Policy)
FDP_IFC.2(2)	Complete information flow control (SCIF Mode Policy)
FDP_IFF.1-NIAP-0407(1)	Simple security attributes (Application Separation Policy)
FDP_IFF.1-NIAP-0407(2)	Simple security attributes (SCIF Mode Policy)
FDP_IFF.1-NIAP-0407(3)	Simple security attributes (External Flow Policy)
FDP_RIP.2	Residual information protection
FDP_UCT.1	Basic data exchange confidentiality
FIA_AFL.1	Authentication failure handling

Functional Components (from CC Part 2)	
FIA_ATD.1	User attribute definition
FIA_UAU.2	User authentication before any action
FIA_UID.2	User identification before any action
FIA_USB.1-NIAP-0415	User-subject binding
FMT_MOF.1(1)	Management of security functions behavior (TSF non-Cryptographic Self-test)
FMT_MOF.1(2)	Management of security functions behavior (Cryptographic Self-tests)
FMT_MOF.1(3)	Management of security functions behavior (Self-tests)
FMT_MTD.1	Management of TSF data
FMT_MTD.2	Management of limits on TSF data
FMT_SMF.1	Specification of management functions
FMT_SMR.2	Restrictions on security roles
FMT_SMR.3	Assuming roles
FPT_RCV.2	Automated recovery
FPT_RPL.1	Replay Detection (External Flow)
FPT_RVM.1	Non-bypassability of the TSP
FPT_SEP.2	SFP domain separation
FPT_STM.1	Reliable time stamps
FPT_TST_(EXP).4	TSF testing (with cryptographic integrity verification)
FPT_TST_(EXP).5	Cryptographic self-test
FRU_RSA.1	Maximum quotas
FTA_SSL.1	TSF-initiated session locking
FTA_SSL.2	User-initiated locking

Functional Components (from CC Part 2)	
FTA_TAB.1	Default TOE access banners
FTP_TRP.1	Trusted path

5.1.1 Security Audit (FAU)

5.1.1.1 FAU_ARP.1 Security alarms

FAU_ARP.1.1 – The TSF shall [immediately display a message identifying the potential security violation, and make accessible the audit record contents associated with the auditable event(s) that generated the alarm, at the :

a) local console;

b) [selection: [ST assignment: other methods determined by the ST author], no other methods]]

upon detection of a potential security violation.

76 *Application Note: The TSF provides a message to the local console regardless of whether an administrator is logged in. The audit records contents associated with the alarm may or may not be part of the message displayed, however the relevant audit information must be available to administrators. In addition, the TOE provides an audible alarm that can be configured to sound an alarm if desired by the Administrator. It is acceptable for the ST author to fill the open assignment with none, if no other methods (e.g., pager, e-mail) are included in the TOE.*

5.1.1.2 FAU_ARP_ACK_(EXP).1 Explicit: Security alarm acknowledgement

FAU_ARP_ACK_(EXP).1.1 – The TSF shall display the alarm message identifying the potential security violation and make accessible the audit record contents associated with the auditable event(s) until it has been acknowledged.

FAU_ARP_ACK_(EXP).1.2 – The TSF shall display an acknowledgement message identifying a reference to the potential security violation, a notice that it has been acknowledged, the time of the acknowledgement and the user identifier that acknowledged the alarm, at the:

local console.

77 *Application Note: This explicit requirement is necessary since a CC requirement does not exist to ensure that the user or administrator will be aware of the alarm.*

The message will not be scrolled off the screen due to other activity-taking place (e.g., the Administrator is running an audit report). Acknowledging the message could be a single event, or different events.

78 *FAU_ARP_ACK_(EXP).1.2 ensures that the user or administrator that received the alarm message also receives the acknowledgement message, which includes some form of reference to the alarm message, who acknowledged the message and when.*

5.1.1.3 FAU_GEN.1-NIAP-0407 Audit data generation

FAU_GEN.1.1-NIAP-0407 – The TSF shall be able to generate an audit record of the following auditable events:

- Start-up and shutdown of the audit functions;
- All auditable events listed in Table 10;
- [selection: [assignment: events at a basic level of audit introduced by the inclusion of additional SFRs determined by the ST author], [assignment: events commensurate with a basic level of audit introduced by the inclusion of explicit requirements determined by the ST author], "no additional events"].

79 *Application Note: For the selection, the ST author should choose one or both of the assignments (as detailed in the following paragraphs), or select "no additional events". For the first assignment, the ST author augments the table (or lists explicitly) the audit events associated with the basic level of audit for any SFRs that the ST author includes that are not included in this PP.*

80 *Likewise, for the second assignment the ST author includes audit events that may arise due to the inclusion of any explicit requirements not already in the PP. Because "basic" audit is not defined for such requirements, the ST author will need to determine a set of events that are commensurate with the type of information that is captured at the basic level for similar requirements.*

81 *If no additional (CC or explicit) SFRs are included, or if additional SFRs are included that do not have "basic" audit associated with them, then it is acceptable to assign "no additional events" in this item.*

FAU_GEN.1.2-NIAP-0407 - The TSF shall record within each audit record at least the following information:

a) Date and time of the event, type of event, subject identity (if applicable), and the outcome (success or failure) of the event; and

b) For each audit event type, based on the auditable event definitions of the functional components included in the PP/ST, [information specified in column three of Table 10 below].

82 *Application Note: In column 3 of the table below, "if applicable" is used to designate data that should be included in the audit record if it "makes sense" in the context of the event that generates the record. If no other information is required (other than that listed in Item a above) for a particular audit event type, then an assignment of "none" is acceptable.*

Table 10 Base Package Auditable Events

Requirement	Auditable Events	Additional Audit Record Contents
FAU_ARP.1	Actions taken due to imminent security violations.	Identification of what caused the generation of the alarm (e.g. process ID)
FAU_ARP_ACK_(EXP).1	None.	
FAU_GEN.1-NIAP-0407	None.	
FAU_GEN.2-NIAP-0410	None.	
FAU_SAA.1-NIAP-0407	a) Enabling and disabling of the analysis mechanisms; b) Automated responses performed by the tool.	
FAU_SAR.1	None.	
FAU_SAR.2	None.	
FAU_SAR.3	None.	
FAU_SEL.1-NIAP-0407	None.	
FAU_STG.1-NIAP-0429	None.	
FAU_STG.3	None.	
FAU_STG.NIAP-0414	None.	
FCS_BCM_(EXP).1	None.	
FCS_CKM.1(1) (Key Generation – Symmetric)	a) Success and failure of the activity. b) The object attribute(s), and object value(s) excluding any sensitive information (e.g. secret or private keys).	
FCS_CKM.1(2) (Key Generation – Asymmetric)	a) Success and failure of the activity. b) The object attribute(s), and object value(s) excluding any	

Requirement	Auditable Events	Additional Audit Record Contents
	sensitive information (e.g. secret or private keys).	
FCS_CKM.2 (Key distribution)	a) Success and failure of the activity. b) The object attribute(s), and object value(s) excluding any sensitive information (e.g. secret or private keys).	
FCS_CKM.4 (Key destruction)	a) Success and failure of the activity. b) The object attribute(s), and object value(s) excluding any sensitive information (e.g. secret or private keys).	
FCS_CKM_(EXP).1 (Key validation/packaging)	a) Success and failure of the activity. b) The object attribute(s), and object value(s) excluding any sensitive information (e.g. secret or private keys).	
FCS_CKM_(EXP).2 (Key handling/storage)	a) Success and failure of the activity. b) The object attribute(s), and object value(s) excluding any sensitive information (e.g. secret or private keys).	
FCS_COA_(EXP).1	None.	
FCS_COP.1(1). (Data encryption/decryption)	a) Success and failure of the operation. b) Any applicable cryptographic modes of operation, subject attributes and object attributes.	
FCS_COP.1(2) (Crypto signature)	a) Success and failure of the operation. b) Any applicable cryptographic modes of operation, subject attributes and	

Requirement	Auditable Events	Additional Audit Record Contents
	object attributes.	
FCS_COP.1(3) (Crypto hashing)	a) Success and failure of the operation. b) Any applicable cryptographic modes of operation, subject attributes and object attributes.	
FCS_COP.1(4) (Crypto Key agreement)	a) Success and failure of the operation. b) Any applicable cryptographic modes of operation, subject attributes and object attributes.	
FCS_COP_(EXP).1 (Random number generation)	Failure of cryptographic operation.	Type of cryptographic operation Any applicable cryptographic mode(s) of operation, excluding any sensitive information
FDP_ACC.2(1)	None.	
FDP_ACC.2(2)	None.	
FDP_ACF.1(1)	Successful requests to perform an operation on an object covered by the SFP. All requests to perform an operation on an object covered by the SFP.	
FDP_ACF.1(2)	Successful requests to perform an operation on an object covered by the SFP. All requests to perform an operation on an object covered by the SFP.	
FDP_IFC.1	None.	
FDP_IFC.2(1) (Application Separation Policy)	None.	
FDP_IFC.2(2) (SCIF Mode Policy)	None.	
FDP_IFF.1-NIAP-0407(1)	All decisions on requests for information Flow.	

Requirement	Auditable Events	Additional Audit Record Contents
(Application Separation Policy)	information Flow.	
FDP_IFF.1-NIAP-0407(2) (SCIF Mode Policy)	All decisions on requests for information Flow.	
FDP_IFF.1-NIAP-0407(3) (External Flow Policy)	All decisions on requests for information Flow.	
FDP_RIP.2	None.	
FDP_UCT.1	The identity of any user or subject using the data exchange mechanisms. The identity of any unauthorized user or subject attempting to use the data exchange mechanisms.	
FIA_AFL.1	The reaching of the threshold for the unsuccessful authentication attempts and the actions (e.g. disabling of a terminal) taken and the subsequent, if appropriate, restoration to the normal state (e.g. re-enabling of a terminal).	Identity of the unsuccessfully authenticated user
FIA_ATD.1	None.	
FIA_UAU.2	All use of the authentication mechanism.	
FIA_UID.2	All use of the user identification mechanism, including the user identity provided.	Claimed identity of the user using the identification mechanism
FIA_USB.1-NIAP-0415 (user-subject binding)	Success and failure of binding of user security attributes to a subject (e.g. success and failure to create a subject).	The identity of the user whose attributes are attempting to be bound
FMT_MOF.1(1)	None.	
FMT_MOF.1(2)	None.	
FMT_MOF.1(3)	None.	
FMT_MTD.1	None.	
FMT_MTD.2	None.	

Requirement	Auditable Events	Additional Audit Record Contents
FMT_SMF.1	None.	
FMT_SMR.2	None.	
FMT_SMR.3	None.	
FPT_RCV.2	a) The fact that a failure or service discontinuity occurred; b) Resumption of the regular operation c) Type of failure or service discontinuity.	
FPT_RPL.1	Detected replay attacks.	Identity of the user that was the subject of the reply attack
FPT_RVM.1	None.	
FPT_SEP.2	None.	
FPT_STM.1	Changes to the time.	
FPT_TST_(EXP).4 (TSF Self Test)	Execution of this set of TSF self tests.	
FPT_TST_(EXP).5 (Crypto Self Test)	Execution of this set of TSF self tests	
FRU_RSA.1	None.	
FTA_SSL.1	a) Locking of an interactive session by the session locking mechanism. b) Successful unlocking of an interactive session. c) Any attempts at unlocking an interactive session.	The identity of the user associated with the session being locked or unlocked
FTA_SSL.2	a) Locking of an interactive session by the session locking mechanism. b) Successful unlocking of an interactive session. c) Any attempts at unlocking an interactive session.	The identity of the user associated with the session being locked or unlocked
FTA_TAB.1	None.	

Requirement	Auditable Events	Additional Audit Record Contents
FTP_TRP.1	a) Failures of the trusted path functions b) Identification of the user associated with all trusted path failures, if available. c) All attempted uses of the trusted path functions. d) Identification of the user associated with all trusted path invocations, if available.	Identification of the claimed user identity

Table 11 Cell Package Auditable Events

Requirement	Auditable Events	Additional Audit Record Contents
FDP_IFC[CELL].1(A)	None	
FDP_IFC[CELL].1(B)	None	
FDP_IFF[CELL].1-NIAP-0407(A)	All decisions on requests for information Flow.	
FDP_IFF[CELL].1-NIAP-0407(B)	All decisions on requests for information Flow.	

5.1.1.4 FAU_GEN.2-NIAP-0410 User identity association

FAU_GEN.2.1-NIAP-0410 For audit events resulting from actions of identified users, the TSF shall be able to associate each auditable event with the identity of the user that caused the event.

5.1.1.5 FAU_SAA.1-NIAP-0407 Potential violation analysis

FAU_SAA.1.1-NIAP-0407 – The TSF shall be able to apply a set of rules in monitoring the audited events and based upon these rules indicate a potential violation of the TSP.

FAU_SAA.1.2-NIAP-0407 - **Refinement**: The TSF shall monitor the accumulation or combination of the following events known to indicate a potential security violation:

a) administrator-specified number of authentication failures;

b) Any detected replay of TSF data or security attributes;

c) Any failure of the cryptographic self-tests;

d) Any failure of the other TSF self-tests;

e) administrator-specified number of encryption failures;

f) administrator-specified number of decryption failures; and

g) [selection: [assignment: additional events from the set of defined auditable events], "no additional events"].

83 *Application Note: The intent of this requirement is that an alarm is generated (FAU_ARP.1) once the threshold for an event is met. Once the alarm has been generated it is assumed that the "count" for that event is reset to zero. The administrator-settable number of authentication failures in (a) is intended to be the same value as specified in FIA_AFL.1.*

84 *The failure of TSF self-tests in (d) include failures of FPT_TST_(EXP)4.*

5.1.1.6 FAU_SAR.1 Audit review

FAU_SAR.1.1 The TSF shall provide [assignment: authorized users] with the capability to read [assignment: list of audit information] from the audit records.

FAU_SAR.1.2 The TSF shall provide the audit records in a manner suitable for the user to interpret the information.

5.1.1.7 FAU_SAR.2 Restricted audit review

FAU_SAR.2.1 The TSF shall prohibit all users read access to the audit records, except those users that have been granted explicit read-access.

5.1.1.8 FAU_SAR.3 Selectable audit review

FAU_SAR.3.1 The TSF shall provide the ability to perform [selection: searches, sorting, ordering] of audit data based on [assignment: criteria with logical relations].

5.1.1.9 FAU_SEL.1-NIAP-0407 Selective Audit

FAU_SEL.1.1-NIAP-0407 - **Refinement**: The TSF shall allow only the administrator to include or exclude auditable events from the set of audited events based on the following attributes:

a) user identity;

b) event type;

c) [selection: object identity, subject identity, host identity, “none”];

d) success of auditable security events;

e) failure of auditable security events; and

f) [selection: [assignment: list of additional criteria that audit selectivity is based upon], no additional criteria]].

85 *Application Note: “event type” is to be defined by the ST author; the intent is to be able to include or exclude classes of audit events.*

5.1.1.10 FAU_STG.1-NIAP-0429 Protected audit trail storage

FAU_STG.1.1-NIAP-0429 – **Refinement**: The TSF shall restrict the deletion of stored audit records in the audit trail to the administrator.

FAU_STG.1.2-NIAP-0429 **Refinement**: The TSF shall be able to prevent modifications to the audit records in the audit trail.

5.1.1.11 FAU_STG.3 Action in case of possible audit data loss

FAU_STG.3.1 - **Refinement**: The TSF shall [immediately alert the administrator and user by displaying a message at the local console, [assignment: other actions determined by the ST author]] if the audit trail exceeds [a administrator-settable percentage of storage capacity].

86 *Application Note: The ST Author should determine if there are other actions that should be taken when the audit trial setting is exceeded, and put these in the assignment. If there are no other actions, then a null assignment is acceptable.*

5.1.1.12 FAU_STG.NIAP-0414 Site-configurable Prevention of audit data loss

FAU_STG.NIAP-0414-1. The TSF shall provide the administrator the capability to select one or more of the following actions [selection: 'ignore auditable events', 'prevent auditable events, except those taken by the authorized user with special rights', 'overwrite the oldest stored audit records'] and [assignment: other actions to be taken in case of audit storage failure] to be taken if the audit trail is full.

FAU_STG.NIAP-0414-2-NIAP-0429 **Refinement**: The TSF shall enforce the administrator’s [selection: choose one of: "ignore auditable events", "prevent auditable events, except those taken by the authorized user with special rights", "overwrite the oldest stored audit records"] and [assignment: other actions to be taken in case of audit storage failure] if the audit trail is full.

87 *Application Note: The TOE provides the administrator the option of preventing audit data loss by preventing auditable events from occurring. The administrator's actions under these circumstances are not required to be audited. The TOE also provides the administrator the option of overwriting "old" audit records rather than preventing auditable events, which may protect against a denial-of-service attack. The ST writer should fill in other technology-specific actions that can be taken for audit storage failure (in addition to the two already specified), or select "no additional options" if there are no such technology-specific actions.*

5.1.2 Cryptographic Support (FCS)[1]

5.1.2.1 FCS_BCM_(EXP).1 Baseline cryptographic module

FCS_BCM_(EXP).1.1 All cryptographic modules shall comply with FIPS PUB 140-2 when performing FIPS-approved cryptographic functions in FIPS-approved cryptographic modes of operation.

FCS_BCM_(EXP).1.2 Cryptographic functions and cryptographic modes of operation as identified in this PP shall be NSA-validated.

88 *Application Note: In time, PED PP cryptographic requirements are expected to evolve such that NSA-validated cryptographic modules shall only contain cryptographic functions, cryptographic modes of operation, and other types of cryptographic processing that are compliant with this protection profile.*

FCS_BCM_(EXP).1.3 All cryptographic modules implemented in the TSF [selection:

- Entirely in hardware shall have a minimum overall rating of FIPS PUB 140-2, Level 3;
- Entirely in software shall have a minimum overall rating of FIPS PUB 140-2, Level 1 and also meet FIPS PUB 140-2, Level 3 for the following: Cryptographic Module Ports and Interfaces; Roles, Services and Authentication; Cryptographic Key Management; Design Assurance; and FIPS PUB 140-2, Level 4 Self Tests[2] as defined by this Protection Profile.
- As a combination of hardware and software shall have a minimum overall rating of FIPS PUB 140-2, Level 1 and also meet FIPS PUB 140-2, Level 3

[1] In drafting specific requirements for this section for general-purpose operating systems, experts were consulted and their input was incorporated. The result is a very minimal set of crypto-related requirements chosen to be consistent with the other requirements of this CC-based protection profile. These crypto requirements are expected to be achievable in commercial products in the near term, and to gradually mature over time.
Evolving public standards on cryptographic functions and related areas have required the following interim approach to writing these cryptographic requirements for general purpose operating systems. This approach uses a variety of footnotes and application notes in an attempt to fill gaps, forewarn of future plans, and/or qualify interpretation of the existing referenced standards (sometimes specific draft versions). As a result, in many instances the presentation of the crypto requirements here is more cumbersome than desired. Still, today these requirements represent a step in the direction of helping to improve the security in COTS products. Over time the approach and presentation will be expanded upon and refined. Correspondingly, the PP will be updated as the underlying public standards and the body of related special publications mature.

[2] Security Level 4 Self Tests comprise the Security Level 1 Self Tests in FIPS PUB 140-2 and the Statistical RNG Tests in Appendix C of this protection profile. These Statistical RNG Tests are the same as those included in the 25 May 2001 version of FIPS PUB 140-2.

for the following: Cryptographic Module Ports and Interfaces; Roles, Services and Authentication; Cryptographic Key Management; Design Assurance; and FIPS PUB 140-2, Level 4 Self Tests[3] as defined by this Protection Profile.]

89 *Application Note: "Combination of hardware and software" means that some part of the cryptographic functionality will be implemented as a software component of the TSF. The combination of a cryptographic hardware module and a software device driver whose sole purpose is to communicate with the hardware module is considered a hardware module rather than a "combination of hardware and software".*

5.1.2.2 FCS_CKM.1(1) Cryptographic key generation

FCS_CKM.1.1(1) **Refinement**: The TSF shall generate[4] **symmetric** cryptographic keys in accordance with a specified cryptographic key generation algorithm as follows: [selection:

- A hardware random number generator (RNG) as specified in FCS_COP_(EXP).1, but with a NIST-approved hashing function required for mixing, and/or

- A software RNG as specified in FCS_COP_(EXP).1]

That meets the following:

- FIPS PUB 180-2, Secure Hash Algorithm

5.1.2.3 FCS_CKM.1(2) Cryptographic key generation

FCS_CKM.1.1(2) **Refinement:** The TSF shall generate[5] **asymmetric[6]** cryptographic keys in accordance with a **domain parameter generator** and ***[selection:***

- a random number generator and/or

- a prime number generator].

that meet the following:

[3] See previous footnote.

[4] This requirement applies strictly to **generation** of symmetric keys. **Validation** techniques for generated symmetric keys are discussed in FCS_CKM_EXP.1.1.

[5] This requirement applies strictly to **generation** of asymmetric keys. **Validation** techniques for generated asymmetric keys are discussed in FCS_CKM_EXP.1.2.

[6] These are the keys/parameters (e.g., the public/private key pairs) underlying a public key-based key establishment scheme, not the session keys established by such schemes.

- Generated key strength shall be equivalent to, or greater than, a symmetric key strength of 128 bits using conservative estimates;
- ANSI X9.80 (3 January 2000), Prime Number Generation, Primality Testing, and Primality Certificates using random integers with deterministic tests, or constructive generation methods;
- Case: For domain parameters used in finite field-based key establishment schemes
    - ANSI X9.42-2001, Public Key Cryptography for the Financial Services Industry: Agreement of Symmetric Keys Using Discrete Logarithm Cryptography; [7]

90 *Application Note: For example, "Classic" Diffie-Hellman-based schemes*

- Case: For domain parameters used in RSA-based key establishment schemes (with odd e)
    - ANSI X9.31-1998 (May 1998), Digital Signatures Using Reversible Public Key Cryptography for the Financial Services Industry (rDSA) for the generation of the RSA parameters[8]; and

91 *Application Note: Although ANSI X9.31 is a standard intended for digital signatures, it is being used here for its coverage of the generation of RSA parameters since ANSI X9.44 is still under development. Once ANSI X9.44 is approved it will be referenced here.*

- Case: For domain parameters used in elliptic curve-based key establishment schemes
    - ANSI X9.63-200x (1 Oct 2000), Public Key Cryptography for the Financial Services Industry: Key Agreement and Key Transport using Elliptic Curve Cryptography. [9]

[7] Any pseudorandom RNG used in these schemes for generating private values shall be seeded by a nondeterministic RNG (both types of RNGs meeting RNG requirements in this PP).

[8] A pseudorandom RNG seeded by a nondeterministic RNG (both types of RNGs meeting RNG requirements in this PP) shall be used in the generation of these primes.

[9] Any pseudorandom RNG used in these schemes for generating private values shall be seeded by a nondeterministic RNG (both types of RNGs meeting RNG requirements in this PP).

5.1.2.4 FCS_CKM.2 Cryptographic Key Distribution[10]

FCS_CKM.2.1 The TSF shall distribute cryptographic keys in accordance with a specified cryptographic key distribution method [selection: Manual (Physical) Method, Automated (Electronic) Method, Manual Method and Automated Method] that meets the following:

a) Manual (Physical) Methods:

- The TSF shall support manual distribution of symmetric keys in accordance with FIPS PUB 171 (Key Management Using ANSI X9.17)[11].

b) Automated (Electronic) Methods:

- The TSF shall automatically distribute symmetric keys in accordance with FIPS PUB 171 (Key Management Using ANSI X9.17).

5.1.2.5 FCS_CKM.4 Cryptographic Key Destruction

FCS_CKM.4.1 **Refinement**: The TSF shall destroy cryptographic keys in accordance with a cryptographic key zeroization method that meets the following:

- FIPS PUB 140-2;
- Zeroization of all plaintext cryptographic keys and all other critical cryptographic security parameters shall be immediate and complete; and
- For embedded cryptographic modules, the zeroization shall be executed by overwriting the key/critical cryptographic security parameter storage area three or more times using a different alternating data pattern each time.

92 *Application Note: Although verification of this zeroization of a plaintext key/critical cryptographic security parameter is desired here (by checking for the final known alternating data pattern), it is not required at this time. However, vendors are highly encouraged to incorporate this verification whenever possible into their implementations.*

[10] Key Distribution (and key establishment) is typically addressed in terms of key transport methods or key agreement methods. Key transport methods are discussed in this section. Key agreement methods are addressed in FCS_COP.1(4) (Cryptographic Operation (for cryptographic key agreement)).

[11] Until NIST identifies approved methods for manually distributing symmetric key, FIPS PUB 171 (Key Management Using ANSI X9.17) shall be used. For purposes of interpreting FIPS PUB 171, only the Triple Data Encryption Algorithm (TDEA) with 168 bits of key shall be applied. (DES is not acceptable for meeting this requirement. Eventual migration to AES is expected.)

93 *Application Note: Zeroization of any storage, such as memory buffers, that is included in the path of a plaintext key/critical cryptographic security parameter is addressed in FCS_CKM_(EXP).2 (Cryptographic Key Handling and Storage).*

5.1.2.6 FCS_CKM_(EXP).1 Cryptographic key validation and packaging

FCS_CKM_(EXP).1.1 The TSF shall apply validation techniques (e.g., parity bits or checkwords) to generated symmetric keys in accordance with:

- FIPS PUB 46-3 (Data Encryption Standard (DES)), and
- FIPS PUB 171[12] (Key Management Using ANSI X9.17).

FCS_CKM_(EXP).1.2 The TSF shall apply validation techniques to generated **asymmetric** keys in accordance with the standards corresponding to the generation technique as called out in FCS_CKM.1.1(2).

FCS_CKM_(EXP).1.3 Any public key certificates generated by the TSF shall be in accordance with NSA-certified NSA-approved certificate schemes[13].

5.1.2.7 FCS_CKM_(EXP).2 Cryptographic key handling and storage

FCS_CKM_(EXP).2.1 The TSF shall perform key entry and output in accordance with FIPS PUB 140-2, Level 4.

FCS_CKM_(EXP).2.2 The TSF shall provide a means to ensure that keys are associated with the correct entities (i.e., person, group, or process) to which the keys are assigned.

FCS_CKM_(EXP).2.3 The TSF shall perform a key error detection check on each transfer of key (internal, intermediate transfers).

94 *Application Note: A parity check is an example of a key error detection check.*

FCS_CKM_(EXP).2.4 The TSF shall encrypt or split persistent secret and private keys when not in use.

95 *Application Note: A persistent key, such as a file encryption key, is one that must be available in the system over long periods of time. A non-persistent key, such as a*

[12] For purposes of interpreting this standard, only TDEA with 168 bits of key shall be applied (DES is not acceptable for meeting this requirement. Eventual migration to AES is expected.).

[13] DoD multilevel applications require Class 5 PKI to address worst case environments, but currently this class is just a concept. In the interim, NSA-approved certificate schemes with hardware tokens for protection of private keys are approved under the added requirement that stronger protection mechanisms must be applied at the boundaries of the protected environment as stated earlier in this PP. When Class 5 certificates are fully established, they will be required.

key used to encrypt or decrypt a single message or a session, is one that is ephemeral in the system.

96 *Application Note: "When not in use" shall be interpreted in the strictest sense so that persistent keys only exist in plaintext form during intervals of operational necessity. For example, a file encryption key shall exist in plaintext form only during actual encryption and/or decryption processing of a file. Once the file is decrypted or encrypted the file encryption key shall be immediately covered for protection.*

FCS_CKM_(EXP).2.5 The TSF shall destroy non-persistent cryptographic keys after an Administrator-defined period of time of inactivity.

FCS_CKM_(EXP).2.6 The TSF shall overwrite each intermediate storage area for plaintext key/critical cryptographic security parameter (i.e., any storage, such as memory buffers, that is included in the path of such data). This overwriting shall be executed three or more times using a different alternating data pattern each time upon the transfer of the key/critical cryptographic security parameter to another location.

97 *Application Note: This is related to the elimination of internal, temporary copies of plaintext keys created during processing, not to the total destruction of a key from the TOE which is discussed under Key Destruction. Although verification of the zeroization of each intermediate location of a plaintext key/critical cryptographic security parameter is desired here (by checking for the final known alternating data pattern), it is not required at this time. However, vendors are highly encouraged to incorporate this verification whenever possible into their implementations.*

5.1.2.8 FCS_COA_(EXP).1 Cryptographic Operations Availability

FCS_COA_(EXP).1 The TSF shall provide the following cryptographic operations to applications:

- Encryption
- Decryption
- Digital Signature
- Key agreement
- Secure hashing
- [assignment: any other cryptographic operations provided to applications].

5.1.2.9 FCS_COP.1(1) Cryptographic operation (for data encryption/decryption)

FCS_COP.1.1(1) **Refinement**: The TSF shall perform **data encryption/decryption services** in accordance with a **NIST-approved implementation of** the cryptographic algorithm **Triple Data Encryption Algorithm (TDEA)**[14] **used in NIST-approved modes of operation** and cryptographic key size **of 168 bits (three independent keys)** that meets the following:

- FIPS PUB 140-2, security Requirements for Cryptographic Modules,
- FIPS PUB 46-3, Data Encryption Standard, and
- ANSI X9.52-1998, Triple Data Encryption Algorithm Modes of Operation.

5.1.2.10 FCS_COP.1(2) Cryptographic operation (Cryptographic signature)

FCS_COP.1.1(2) **Refinement:** The TSF shall perform **cryptographic signature services** in accordance with the **NIST-approved digital signature algorithm** [selection:

- Digital Signature Algorithm (DSA) with a key size (modulus) of 2048[15] bits or greater,
- RSA Digital Signature Algorithm (rDSA with odd e) with a key size (modulus) of 2048[16] bits or greater, or
- Elliptic Curve Digital Signature Algorithm (ECDSA) with a key size of 256 bits or greater].

98 *Application Note: For elliptic curve-based schemes the key size refers to the log_2 of the order of the base point. As the preferred approach for cryptographic signature, elliptic curves will be required within a TBD time frame after all the necessary standards and other supporting information are fully established.*

[14] The Advanced Encryption Standard (AES) employing key lengths of 128 bits or greater and meeting NIST-approved AES standards will be required when AES is fully established. With the approval of FIPS PUB 197 and NIST Special Publication 800-38A, progress is being made to fully establish AES, but establishment is not yet complete. Other approved public standards or NIST special publications are still needed for AES. (An example of this is key distribution for AES.)

[15] A 2048-bit or greater modulus is required to provide the desired 128-bit equivalent symmetric key strength. The 2048-bit modulus is compatible with (1.) operationally practical digital signature key sizes in pending IPSEC commercial products, and (2.) the current direction of digital signatures in the DoD PKI. This smaller modulus reduces the equivalent symmetric key strength to 112 bits. Certificate signatures based on a 2048-bit or greater modulus or the elliptic curve approach is recommended as soon as the DoD PKI can support it. The elliptic curve approach is preferred. *{"Near term applications" means products designed and validated against this specific version of this PP.}*

[16] See previous footnote.

that meet the following:

- Case: Digital Signature Algorithm

  FIPS PUB 186-2[17], Digital Signature Standard, for signature creation and verification processing; and ANSI Standard X9.42-2001, Public Key Cryptography for the Financial Services Industry: Agreement of Symmetric Keys Using Discrete Logarithm Cryptography for generation of the domain parameters[18];

- Case: RSA Digital Signature Algorithm (with odd e)

  .ANSI X 9.31-1998 (May 1998), Digital Signatures Using Reversible Public Key Cryptography For The Financial Services Industry (rDSA)[19];

- Case: Elliptic Curve Digital Signature Algorithm

  ANSI X9.62-1-xxxx (10 Oct 1999), Public Key Cryptography for the Financial Services Industry: Elliptic Curve Digital Signature Algorithm (ECDSA) **[20]**.

5.1.2.11 FCS_COP.1(3) Cryptographic Operation (Cryptographic hashing)

FCS_COP.1.1(3) **Refinement:** The TSF shall perform cryptographic hashing services in accordance with a **NIST-approved hash implementation of the Secure Hash algorithm and message digest size of at least 256 bits** that meets the following: FIPS PUB 180-2.

99 *Application Note: The message digest size should correspond to double the system encryption key strength.*

5.1.2.12 FCS_COP.1(4) Cryptographic Operation (Cryptographic key agreement)

FCS_COP.1.1(G2) **Refinement:** The TSF shall perform **cryptographic key agreement services** in accordance with a **NIST-approved implementation of a key agreement [21]** algorithm *[selection:*

[17] FIPS PUB 186-3 is under development. It will incorporate the signature creation and verification processing of FIPS PUB 186-2, and the generation of domain parameters of ANSI X9.42. FIPS PUB 186-3 shall be used here when it is finalized and approved.

[18] Any pseudorandom RNG used in these schemes for generating private values shall be seeded by a nondeterministic RNG (both types of RNGs meeting RNG requirements in this PP).

[19] See previous footnote.

[20] See previous footnote.

[21] Until FIPS PUB 140-2 identifies approved key agreement schemes, NIST Special Publication 800-56 ("Recommendation on Key Establishment Schemes", DRAFT 2.0, Jan 2003) shall be used here.

- Finite Field-based key agreement algorithm and cryptographic key sizes(modulus) of 2048 bits or greater,
- Elliptic Curve-based key agreement algorithm and cryptographic key size of 256 bits or greater]

100 *Application Note: For elliptic curve-based schemes the key size refers to the $\log_2$ of the order of the base point. As the preferred approach for key exchange, elliptic curves will be required within a TBD time frame after all the necessary standards and other supporting information are fully established.*

That meets the following:

- Case: Finite field-based key agreement schemes

  ANSI X9.42-2001, Public Key Cryptography for the Financial Services Industry: Agreement of Symmetric Keys Using Discrete Logarithm Cryptography[22]:

101 *Application Note: For example, "Classic" Diffie-Hellman-based schemes.*

- Case: Elliptic curve-based key agreement schemes

  ANSI X9.63-200x (1 Oct 2000), Public Key Cryptography for the Financial Services Industry: Key Agreement and Key Transport using Elliptic Curve Cryptography. [23]

102 *Application Note: Some authentication mechanism on the keying material is recommended. In addition, repeated generation of the same shared secrets should be avoided. As an example, the MQV schemes described in the above standards address these issues.*

5.1.2.13 FCS_COP_(EXP).1 Random Number Generation

FCS_COP_(EXP).1.1 The TSF shall perform all random number generation (RNG) services in accordance with [selection:

- Multiple independent hardware-generated inputs combined with a mixing function, or

103 *Application Note: A NIST-approved hashing function is recommended for the mixing function in hardware based RNGs. If the length of the needed random number exceeds the length of the hash's message digest, then multiple hashes can be used to prove the needed random quantity.*

[22] Any pseudorandom RNG used in these schemes for generating private values shall be seeded by a nondeterministic RNG (both types of RNGs meeting RNG requirements in this PP).

[23] See previous footnote.

- Multiple independent software-generated inputs combined with a NIST-approved hashing function, or

104 *Application Note: A NIST-approved hashing function is required for the mixing function in software based RNGs. If the length of the needed random number exceeds the length of the hash's message digest, then multiple hashes can be used to prove the needed random quantity.*

- A combination of multiple independent hardware-generated inputs combined with a mixing function and multiple independent software-generated inputs combined with a NIST-approved hashing function]

that meet the following:

- FIPS PUB 180-2, when using a NIST-approved hashing function as the mixing function,

- Documents listed in Section 13 of this PP and NIST Special Publication 800-22: A statistical Test Suite for Random and Pseudorandom Number Generators for cryptographic Applications;

105 *Application Note: This publication includes some discussion and guidance on randomness and RNG seeding. Successful completion and documentation of these tests during the TOE development helps to demonstrate the random number generator design is rigorous. There exists a NIST toolbox for running these tests. Requirements for acceptable thresholds and sample sizes for use in applying NIST Special Publication 800-2 in the context of this protection profile can be found in Section 14 of this profile.*

- All the RNG/PRNG self-tests of FIPS PUB 140-2,

- All statistical RNG tests (as specified in Section 13 of this PP) upon demand and upon power-up,

- The augmented tests, and self-test requirements from this PP: TSF Self Testing, and

- RNG/PRNG design and test documentation consistent with that required in this PP for other subsystems: Development Documentation (ADV).

FCS_COP_(EXP).1.2 The TSF shall defend against tampering of the random number generation (RNG)/pseudorandom number generation (PRNG) sources.

106 *Application Note: The RNG/PRNG should be resistant to manipulation or analysis of its sources, or any attempts to predictably influence its states. Three examples of very different approaches the TSF might pursue to address this include: a) identifying the fact that physical security must be applied to the product, b) applying checksums over the sources, or c) designing and implementing the TSF RNG with a*

concept similar to a keyed hash (e.g., where periodically, the initial state of the hash is changed unpredictably and each change is protected as when provided on a tamper-protected token, or in a secure area of memory.

5.1.3 User Data Protection (FDP)

5.1.3.1 FDP_ACC.2(1) Complete access control

FDP_ACC.2.1(1) The TSF shall enforce the [Stored Data Policy] on

- [Subjects: processes storing and retrieving data in persistent memory,
- Objects: data repositories containing user data]

and all operations among subjects and objects covered by the SFP.

FDP_ACC.2.2(1) The TSF shall ensure that all operations between any subject in the TSC and any object within the TSC are covered by an access control SFP.

5.1.3.2 FDP_ACC.2(2) Complete access control

FDP_ACC.2.1(2) The TSF shall enforce the [Application Access Control Policy] on

- [Subjects: application,
- Objects: application resource]

and all operations among subjects and objects covered by the SFP.

107 *Application Note: The "application resources" are meant to cover the files, directories, and other electronic storage resources used by an application.*

FDP_ACC.2.2(2) The TSF shall ensure that all operations between any subject in the TSC and any object within the TSC are covered by an access control SFP.

5.1.3.3 FDP_ACF.1(1) Security attribute based access control

108 *Interp Note: The following element has changed as a result of Interpretation 103.*

FDP_ACF.1.1(1) The TSF shall enforce the [Stored Data Policy] to objects based on the following:

- [Subjects: symmetric cryptographic keys,
- Objects: none].

FDP_ACF.1.2(1) The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed:

- [All subjects must use symmetric keys to encrypt data being stored in objects using TDEA with a key size of 168 bits (FCS_COP.1(1)),
- All subjects must use symmetric keys to decrypt data being retrieved from objects using TDEA with a key size of 168 bits (FCS_COP.1(1))].

FDP_ACF.1.3(1) The TSF shall explicitly authorize access of subjects to objects based on the following additional rules: [none].

FDP_ACF.1.4(1) The TSF shall explicitly deny access of subjects to objects based on the [none].

5.1.3.4 FDP_ACF.1(2) Security attribute based access control

109 *Interp Note: The following element has changed as a result of Interpretation 103.*

FDP_ACF.1.1(2) The TSF shall enforce the [Application Access Control Policy] to objects based on the following:

- [Subject security attribute: application ownership identifier,
- Object security attribute: application ownership identifier].

FDP_ACF.1.2(2) The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed:

- [An object may only have one subject security attribute associated with it
- A subject may only read/write/execute an object when the subject security attribute equals the object security attribute].

FDP_ACF.1.3(2) The TSF shall explicitly authorize access of subjects to objects based on the following additional rules: [none].

FDP_ACF.1.4(2) The TSF shall explicitly deny access of subjects to objects based on the [none].

5.1.3.5 FDP_IFC.1 Subset information flow

FDP_IFC.1.1 The TSF shall enforce the [External Flow Policy] on:

a) Subjects: TOE hosted applications and external TOE peripherals

b) Information: user data

c) Operations:

- Wireless Access Point (AP) connection
- Web-site session flows;
- Sending and receiving of plain text and S/MIME e-mails
- [assignment: other operations specified in the Security Target.]

110 *Application Note: The synchronization operation is meant to model the synchronization between PED to PED or PED to a host workstation.*

5.1.3.6 FDP_IFC.2(1) Complete information flow (Application Separation Policy)

FDP_IFC.2.1(1) The TSF shall enforce the [Application Separation Policy] on:

- [Subjects: application programs executing on behalf of the authorized user and
- Information: application information]

and all operations that cause information to flow to and from subjects covered by the SFP.

FDP_IFC.2.2(1) The TSF shall ensure that all operations that cause any information in the TSC to flow to and from any subject in the TSC are covered by an information flow control SFP.

5.1.3.7 FDP_IFC.2(2) Complete information flow (SCIF Mode policy)

FDP_IFC.2.1(2) The TSF shall enforce the [SCIF Mode Policy] on:

- [subjects: TOE external interfaces
- information: any external communication protocols, services, and data provided by the TOE’s external interfaces]

and all operations that cause information to flow to and from subjects covered by the SFP.

FDP_IFC.2.2(2) The TSF shall ensure that all operations that cause any information in the TSC to flow to and from any subject in the TSC are covered by an information flow control SFP.

5.1.3.8 FDP_IFF.1-NIAP-0407(1) Simple security attributes (Application Separation Policy)

111 *Interp Note: The following element is changed as a result of Interpretation 104.*

FDP_IFF.1.1-NIAP-0407(1) The TSF shall enforce the [Application Separation Policy] based on the followings types of subjects and information security attributes:

- [Subject attributes: subject identity, subject identifier
- Information security attributes for objects: object identity, object identifier].

FDP_IFF.1.2-NIAP-0407(1) The TSF shall permit an information flow between a controlled subject and controlled information via a controlled operation if the following rules hold:

- [Information flow from object to subject: The subject shall be permitted to access the object for reading, and the domain of the subject shall be permitted to read objects from the domain of the object.
- Information flow from subject to object: The subject shall be permitted to access the object for writing, and the domain of the subject shall be permitted to write data to the domain of the object].

FDP_IFF.1.3-NIAP-0407(1) The TSF shall enforce the following information flow control rules: [selection: [assignment: additional information flow control SFP rules], "no additional information flow control SFP rules"].

FDP_IFF.1.4-NIAP-0407(1) The TSF shall provide the following [selection: [assignment: list of additional SFP capabilities], “no additional SFP capabilities”].

FDP_IFF.1.5-NIAP-0407(1) The TSF shall explicitly authorize an information flow based on the following rules: [selection: [assignment: rules, based on security attributes, that explicitly authorize information flows], “no explicit authorization rules”].

FDP_IFF.1.6-NIAP-0407(1) The TSF shall explicitly deny an information flow based on the following rules: [selection: [assignment: rules, based on security attributes, that explicitly deny information flows], “no explicit denial rules”].

5.1.3.9 FDP_IFF.1-NIAP-0407(2) Simple security attributes (SCIF Mode Policy)

112 *Interp Note: The following element is changed as a result of Interpretation 104.*

FDP_IFF.1.1-NIAP-0407(2) The TSF shall enforce the [SCIF Mode Policy] based on the followings types of subjects and information security attributes:

- [subject attributes: subject identity].

FDP_IFF.1.2-NIAP-0407(2) The TSF shall permit an information flow between a controlled subject and controlled information via a controlled operation if the following rules hold:

- [When the TSF is in SCIF mode all subjects are disabled].

FDP_IFF.1.3-NIAP-0407(2) The TSF shall enforce the following information flow control rules: *no additional information flow control SFP rules*.

FDP_IFF.1.4-NIAP-0407(2) The TSF shall provide the following: *no additional SFP capabilities*.

FDP_IFF.1.5-NIAP-0407(2) The TSF shall explicitly authorize an information flow based on the following rules: *no explicit authorization rules*.

FDP_IFF.1.6-NIAP-0407(2) The TSF shall explicitly deny an information flow based on the following rules: *no explicit denial rules*.

5.1.3.10 FDP_IFF.1-NIAP-0407(3) Simple security attributes (External Flow Policy)

113 *Interp Note: The following element is changed as a result of Interpretation 104.*

FDP_IFF.1-NIAP-0407(3) The TSF shall enforce the [External Flow Policy] based on the following types of subjects and information security attributes:

a) Subject security attributes:

i. X.509 certificate

ii. [selection:[assignment: other subject security attributes determined by the ST author], none]

b) Information security attributes

i. User and session identification

FDP_IFF.1.2-NIAP-0407(3) The TSF shall permit an information flow between a controlled subject and controlled information flow operation if the following rules hold:

i. [Wireless flow rule: The identity of the access point is in the set of access point identifiers

ii. Web traffic flow rule: The identity of the remote server is in the set of secure remote server identifiers

iii. E-mail flow rule: The identity of the user is in the set of user identities with valid X.509 certificates]

FDP_IFF.1.3-NIAP-0407(3) The TSF shall enforce the following information flow control rules: [selection*:[assignment: list of additional rules], "no additional rules"*]

FDP_IFF.1.4-NIAP-0407(3) The TSF shall provide the following: [selection: [assignment: list of additional SFP capabilities], "no additional SFP capabilities"]

FDP_IFF.1.5-NIAP-0407(3) The TSF shall explicitly authorize an information flow based on the following rules: [selection: [assignment: rules, based on security attributes, that explicitly authorize information flows], "no explicit authorization rules"]

FDP_IFF.1.6-NIAP-0407(3) The TSF shall explicitly deny information flow based on the following rules: [selection: assignment: rules, based on security attributes that explicitly deny information flows], "no additional explicit denial rules"]

5.1.3.11 FDP_RIP.2 Full residual information protection

FDP_RIP.2.1 The TSF shall ensure that any previous information content of a resource is made unavailable upon the [selection: deallocation of the resource from] all objects.

5.1.3.12 FDP_UCT.1 Basic data exchange confidentiality

FDP_UCT.1.1 **Refinement**: The TSF shall enforce the [External Flow Policy] to be able to *transmit* **and** *receive* objects in a manner protected from unauthorized disclosure.

5.1.4 Identification and Authentication (FIA)

5.1.4.1 FIA_AFL.1 Authentication failure handling

114 *Interp Note: The following element is changed as a result of Interpretation 111.*

FIA_AFL.1.1 The TSF shall detect when *a Administrator configurable positive integer within [assignment: range of acceptable Administrator configurable amount of time]* unsuccessful authentication attempts occur related to [user authentication].

FIA_AFL.1.2 When the defined number of unsuccessful authentication attempts has been met or surpassed, the TSF shall [lock the device for a Administrator configurable amount of time].

5.1.4.2 FIA_ATD.1 User attribute definition

FIA_ATD.1.1 The TSF shall maintain the following list of security attributes belonging to individual users:

a) [unique user identity,

b) all roles listed in FMT_SMR.2, and

c) [selection: [assignment: list of other security attributes], "no additional security attributes"]].

5.1.4.3 FIA_UAU.2 User authentication before any action

FIA_UAU.2.1 The TSF shall require each user to be successfully authenticated before allowing any other TSF-mediated actions on behalf of that user.

5.1.4.4 FIA_UID.2 User identification before any action

FIA_UID.2.1 The TSF shall require each user to identify itself before allowing any other TSF-mediated actions on behalf of that user.

5.1.4.5 FIA_USB.1-NIAP-0415 User-subject binding

FIA_USB.1.1 The TSF shall associate all user security attributes with subjects acting on the behalf of that user: [all user security attributes listed in FIA_ATD.1].

5.1.5 Security Management (FMT)

5.1.5.1 FMT_MOF.1(1) Management of security functions behaviour (TSF non-Cryptographic Self-test)

FMT_MOF.1.1(1) The TSF shall restrict the ability to *modify the behavior of* the functions [TSF non-Cryptographic Self-test (FPT_TST_(EXP).4)] to [the Administrator].

115 *Application Note: "Modify the behaviour" refers to specifying the interval at which the test periodically runs, or perhaps selecting a subset of the tests to run.*

5.1.5.2 FMT_MOF.1(2) Management of security functions behaviour (Cryptographic Self-tests)

FMT_MOF.1.1(2) The TSF shall restrict the ability to *disable, enable and modify the behavior of* the functions [TSF Cryptographic Self-tests (FPT_TST_(EXP).5)] to [the Administrator].

116 *Application Note: The enabling or disabling of the cryptographic self-tests refers to immediately after key generation. "Modify the behaviour" refers to specifying the interval at which the test periodically runs, or perhaps selecting a subset of the tests to run.*

5.1.5.3 FMT_MOF.1(3) Management of security functions behaviour (Self-tests)

FMT_MOF.1.1(3) The TSF shall restrict the ability to *modify the behavior of* the functions [TSF Self-tests (FPT_TST_(EXP).4 and FPT_TST_(EXP).5)] to [the administrator role].

5.1.5.4 FMT_MTD.1 Management of TSF data

FMT_MTD.1.1 The TSF shall restrict the ability to *change_default, query, modify, delete, clear*, [selection: [assignment: other operations], "take no operation on"] all the [all TSF data] to [Administrator]..

5.1.5.5 FMT_MTD.2 Management of limits on TSF data

FMT_MTD.2.1 The TSF shall restrict the specification of the limits for [assignment: *list of TSFdata*] to [Administrator].

FMT_MTD.2.2 The TSF shall take the following actions, if the TSF data are at, or exceed, the indicated limits: [assignment: *actions to be taken*].

5.1.5.6 FMT_SMF.1 Specification of management functions

117 *Interp Note: This requirement was created as a result of Interpretation 065.*

FMT_SMF.1.1 The TSF shall be capable of performing the following security management functions:

- [Non-Cryptographic Self-Tests (FPT_TST_(EXP).4
- Cryptographic Self-Tests (FPT_TST_(EXP).5

- [assignment: additional security management functions to be provided by the TSF]].

5.1.5.7 FMT_SMR.2 Restrictions on security roles

FMT_SMR.2.1 The TSF shall maintain the roles:

- [Administrator
- [selection: [assignment: any other roles], "none"]].

FMT_SMR.2.2 The TSF shall be able to associate users with roles.

FMT_SMR.2.3 The TSF shall ensure that the conditions:

- [All roles shall be able to administer the TOE locally;
- All roles are distinct; that is, there shall be no overlap of operations performed by each role, with the following exceptions:
    i. All administrators can invoke the self-tests.

are satisfied.

118 *Application Note: The administering of the TOE is limited to the capabilities associated with an administrative role.*

5.1.5.8 FMT_SMR.3 Assuming roles

FMT_SMR.3.1 The TSF shall require an explicit request to assume the following roles: [all roles listed in FMT_SMR.2.1].

5.1.6 Protection of the TOE Security Functions (FPT)

5.1.6.1 FPT_RCV.2 Automated recovery

119 *Interp Note: The following elements changed as a result of Interpretation 056.*

FPT_RCV.2.1 When automated recovery from [assignment: list of failures/service discontinuities] is not possible, the TSF shall enter a maintenance mode where the ability to return to a secure state is provided.

FPT_RCV.2.2 For [assignment: list of failures/service discontinuities], the TSF shall ensure the return of the TOE to a secure state using automated procedures.

5.1.6.2 FPT_RPL.1 Replay detection

FPT_ RPL.1.1 The TSF shall detect replay for the following entities: **encrypted wireless transmissions, authentication data, TSF data and security attributes**.

FPT_ RPL.1.2 The TSF shall perform **data rejection** when replay is detected.

5.1.6.3 FPT_RVM.1 Non-bypassability of the TSP

FPT_RVM.1.1 The TSF shall ensure that TSP enforcement functions are invoked and succeed before each function within the TSC is allowed to proceed.

120 *Application Note: The TOE must provide a security architecture such that all the functionality described by the TOE requirements in this PP cannot be bypassed. This means that the TOE should not have any external interfaces that can bypass the functionality described.*

5.1.6.4 FPT_SEP.2 SFP domain separation

FPT_SEP.2.1 The unisolated portion of the TSF shall maintain a security domain for its own execution that protects it from interference and tampering by untrusted subjects.

FPT_SEP.2.2 The TSF shall enforce separation between the security domains of subjects in the TSC.

FPT_SEP.2.3 The TSF shall maintain the part of the TSF related to [Application Separation Policy] in a security domain for their own execution that protects them from interference and tampering by the remainder of the TSF and by subjects untrusted with respect to those SFPs.

5.1.6.5 FPT_STM.1 Reliable time stamps

FPT_STM.1.1 The TSF shall be able to provide reliable time stamps for its own use.

5.1.6.6 FPT_TST_(EXP).4 TSF testing (with cryptographic integrity verification)

FPT_TST_(EXP).4.1 The TSF shall run a suite of self-tests during initial start-up, periodically during normal operation as specified by the Administrator, and at the request of any administrator to demonstrate the correct operation of the hardware portions of the TSF.

FPT_TST_(EXP).4.2 The TSF shall provide a Administrator with the capability to use a TSF-provided cryptographic function to verify the integrity of all TSF data except the following: passwords, [selection: [assignment: other dynamic TSF data for which no integrity validation is justified], “none”]].

FPT_TST_(EXP).4.3 The TSF shall provide a Administrator with the capability to use a TSF-provided cryptographic function to verify the integrity of stored TSF executable code.

121 *Application Note: This explicit requirement is necessary since some TOE data are dynamic (e.g., passwords) and so interpretation of “integrity” for FPT_TST.1.2 is required, leading to potential inconsistencies. The intention is that any parameter that only an administrator can control is verified to ensure its integrity is maintained. It is not necessary for the TOE to verify the integrity of user’s passwords.*

122 *Since this TOE includes all the hardware necessary for the operation of the TOE, the element FPT_TST_(EXP).4.1 ensures that the hardware aspects of the TOE are tested prior to or during operations. It is not necessary to test the software portions of the TSF, since the evaluation ensures that correct operation of the software, software does not degrade or suffer intermittent faults, as does hardware, and integrity of the software portions of the TSF are addressed by FPT_TST_(EXP).4.3. Note that since cryptographic functions implemented in hardware that are part of a cryptomodule are tested in FPT_TST_(EXP).5, this requirement only applies to cryptographic functionality implemented in hardware that is not implemented in a cryptomodule (for instance, and implementation of a Key agreement algorithm).*

123 *In element FPT_TST_(EXP).4.2, the ST author should specify the TSF data for which integrity validation is not required. While some TSF data are dynamic and therefore not amenable to integrity verification, it is expected that all TSF data for which integrity verification “makes sense” be subject to this requirement.*

124 *In elements FPT_TST_(EXP).4.2 and FPT_TST_(EXP).4.3, the cryptographic mechanism can be any one of the ones specified in FCS_COP.1(2) or FCS_COP_(EXP).1, although typically hash functions or digital signatures are used for integrity verification.*

5.1.6.7 FPT_TST_(EXP).5 Cryptographic self-test

FPT_TST_(EXP).5.1 The TSF shall run the suite of self-tests provided by the FIPS 140-2 cryptographic module during initial start-up (power on), at the request of any administrator, periodically (at a Administrator-specified interval not less than at least once a day) to demonstrate the correct operation of the cryptographic components of the TSF.

FPT_TST_(EXP).5.2 The TSF shall be able to run the suite of self-tests provided by the FIPS 140-2 cryptographic module immediately after the generation of a key.

125 *Application Note: For element FPT_TST_(EXP).5.1, the Administrator has the ability to enable and disable this capability; this is specified in FMT_MOF.1(2).*

5.1.7 Resource Allocation (FRU)

5.1.7.1 FRU_RSA.1 Maximum quotas

FRU_RSA.1.1 The TSF shall enforce maximum quotas of the following resources: [all controlled resources] that *individual user* can use *simultaneously*.

5.1.8 TOE Access (FTA)

5.1.8.1 FTA_SSL.1 TSF-initiated session locking

FTA_SSL.1.1 The TSF shall lock an interactive session after [a Administrator-specified time period of inactivity] by:

a) clearing or overwriting display devices, making the current contents unreadable;

b) disabling any activity of the user's data access/display devices other than unlocking the session.

FTA_SSL.1.2 The TSF shall require the following events to occur prior to unlocking the session: [user authentication].

5.1.8.2 FTA_SSL.2 User-initiated session locking

FTA_SSL.2.1 The TSF shall allow user-initiated locking of the user's own interactive session, by:

a) clearing or overwriting display devices, making the current contents unreadable;

b) disabling any activity of the user's data access/display devices other than unlocking the session.

FTA_SSL.2.2 The TSF shall require the following events to occur prior to unlocking the session: [user authentication].

5.1.8.3 FTA_TAB.1 Default TOE access banners

FTA_TAB.1.1 **Refinement**: Before establishing a user session **that requires authentication**, the TSF shall display **only** a **Administrator-specified** advisory **notice and consent** warning message regarding unauthorized use of the TOE.

126 *Application Note: The access banner applies whenever the TOE will provide a prompt for identification and authentication. The intent of this requirement is to advise users of warnings regarding the unauthorized use of the TOE and to provide the Administrator with control over what is displayed (e.g., if the Administrator chooses, they can remove banner information that informs the user of the product and version number).*

5.1.9 Trusted Path/Channels

5.1.9.1 FTP_TRP.1 Trusted path

FTP_TRP.1.1 The TSF shall provide a communication path between itself and *local* users that is logically distinct from other communication paths and provides assured identification of its end points and protection of the communicated data from modification or disclosure.

FTP_TRP.1.2 The TSF shall permit *local users* to initiate communication via the trusted path.

FTP_TRP.1.3 The TSF shall require the use of the trusted path for *initial user authentication*.

5.2 Cellular Communications Package

The requirements listed here must be combined with the Requirements Table in the base PED PP to achieve a completed mapping. Only those requirements applicable to the cellular functions are listed in this table.

Table 12 Cellular Security Functional Requirements

Functional Components (from CC Part 2)	
FDP_IFC[CELL].1(A)	Subset information flow (Voice Transport Protection Policy)
FDP_IFC[CELL].1(B)	Subset information flow (SMS Transport Protection Policy)
FDP_IFF[CELL].1-NIAP-0407(A)	Simple security attributes (Voice Transport Protection Policy)
FDP_IFF[CELL].1-NIAP-0407(B)	Simple security attributes (SMS Transport Protection Policy)

5.2.1.1 Subset information flow (FDP_IFC[CELL].1(A)) (Voice Transport Protection Policy)

FDP_IFC[CELL].1(A) The TSF shall enforce the [Voice Transport Protection Policy] on:

- Subjects: processes sending and receiving voice data
- Information: voice data
- Operations:
  1. Send/ receive voice data without signing or encryption
  2. Send encrypted and signed voice data
  3. Decrypt and verify signature for received voice data
  4. [assignment: other operations specified in the Security Target.]

5.2.1.2 Subset information flow FDP_IFC[CELL].1(B) (SMS Transport Protection Policy)

FDP_IFC[CELL].1(B) The TSF shall enforce the [SMS Transport Protection Policy] on:

a) Subjects: processes exchanging SMS messages with entities (i.e. servers, other PEDs, phones)

b) Information: user data (i.e., SMS messages)

c) Operations:

- Pass data without signing or encryption
- Send encrypted and signed messages to an entity
- Decrypt and verify signature for received messages from an entity
- [assignment: other operations specified in the Security Target.]

5.2.1.3 Simple security attributes (FDP_IFF[CELL].1-NIAP-0407(A)) (Voice Transport Protection Policy)

FDP_IFF[CELL].1-NIAP-0407(A) The TSF shall enforce the [Voice Transport Protection Policy] based on the following types of subjects and information security attributes:

a) Subject security attributes:

- Set of receiver identities which have valid X.509 certificates
- [selection*:[assignment: other subject security attributes determined by the ST author], none*]

c) Information security attributes

- Identity of the receiver as indicated in their X.509 certificate

FDP_IFF[CELL].1.2-NIAP-0407(A) The TSF shall permit an information flow between a controlled subject and controlled information flow operation if the following rules hold:

- [The identity of the receiver is in the set of receiver identities]

FDP_IFF[CELL].1.3-NIAP-0407(A) The TSF shall enforce the following information flow control rules: [selection: [assignment: list of additional rules], "no additional rules"]

FDP_IFF[CELL].1.4-NIAP-0407(A) The TSF shall provide the following: [selection: [assignment: list of additional SFP capabilities], "no additional SFP capabilities"]

FDP_IFF[CELL].1.5-NIAP-0407(A) The TSF shall explicitly authorize an information flow based on the following rules: [selection: [assignment: rules, based on security attributes, that explicitly authorize information flows], "no explicit authorization rules"]

FDP_IFF[CELL].1.6-NIAP-0407(A) The TSF shall explicitly deny information flow based on the following rules: [selection: assignment: rules, based on security attributes that explicitly deny information flows], "no additional explicit denial rules"]

5.2.1.4 Simple security attributes FDP_IFF[CELL].1-NIAP-0407(B) (SMS Transport Protection Policy)

FDP_IFF[CELL].1-NIAP-0407(B) The TSF shall enforce the [SMS Transport Protection Policy] based on the following types of subjects and information security attributes:

d) Subject security attributes:

- Set of entity identifiers that have valid X.509 certificates
- [selection*:[assignment: other subject security attributes determined by the ST author], none*]

d) Information security attributes

- Identity of the entity as indicated in their X.509 certificate

FDP_IFF[CELL].1.2-NIAP-0407(B) The TSF shall permit an information flow between a controlled subject and controlled information flow operation if the following rules hold:

- [The identity of the entity is in the set of entity identifiers with valid X.509 certificates]

FDP_IFF[CELL].1.3-NIAP-0407(B) The TSF shall enforce the following information flow control rules: [selection: [assignment: list of additional rules], “no additional rules”].

FDP_IFF[CELL].1.4-NIAP-0407(B) The TSF sha ll provide the following: [selection: [assignment: list of additional SFP capabilities], “no additional SFP capabilities”]

FDP_IFF[CELL].1.5-NIAP-0407(B) The TSF shall explicitly authorize an information flow based on the following rules: [selection: [assignment: rules, based on security attributes, that explicitly authorize information flows], “no explicit authorization rules”]

FDP_IFF[CELL].1.6-NIAP-0407(B) The TSF shall explicitly deny information flow based on the following rules: [selection: assignment: rules, based on security attributes that explicitly deny information flows], “no additional explicit denial rules”]

5.3 Security Requirements for the IT Environment

127 This PP does not contain any security requirements for the IT environment.

5.4 TOE Security Assurance Requirements

128 The TOE assurance requirements for this PP no longer map to a CC EAL in accordance with Medium Robustness for Environments Guidance dated 1 March 2004. The assurance requirements are summarized in the Table 13 below. The objectives and application notes for the explicit ADV requirements are contained in Appendix E. The methodology for performing the evaluation activities pertaining to the explicit assurance requirements is provided by CCEVS management in a separate document.

Table 13 Assurance Requirements

Assurance Class	Assurance Components	
Configuration Management	ACM_AUT.1	Partial CM automation
	ACM_CAP.4	Generation support and acceptance procedures
	ADM_SCP.2	Problem tracking CM coverage
Delivery and Operation	ADO_DEL.2	Detection of modification
	ADO_IGS.1	Installation, generation, and start-up procedures
Development	ADV_ARC_(EXP).1	Architectural design with justification
	ADV_FSP_(EXP).1	Functional specification with complete summary
	ADV_HLD_(EXP).1	Security-enforcing high-level design
	ADV_IMP.2	Implementation of the TSF
	ADV_INT_(EXP).1	Modular decomposition
	ADV_LLD_(EXP).1	Security-enforcing low-level design

Assurance Class	Assurance Components	
	ADV_RCR.1	Informal correspondence demonstration
	ADV_SPM.1	Informal TOE security policy model
Guidance Documents	AGD_ADM.1	Administrator guidance
	AGD_USR.1	User guidance
Life Cycle Support	ALC_DVS.1	Identification of security measures
	ALC_FLR.2	Flaw reporting procedures
	ALC_LCD.1	Developer defined life-cycle model
	ALC_TAT.1	Well-defined development tools
Tests	ATE_COV.2	Analysis of coverage
	ATE_DPT.2	Testing: low-level design
	ATE_FUN.1	Functional testing
	ATE_IND.2	Independent testing – sample
Vulnerability Assessment	AVA_CCA_(EXP).2	Systematic cryptographic module covert channel analysis
	AVA_MSU.2	Validation of analysis
	AVA_SOF.1	Strength of TOE security functional evaluation
	AVA_VLA.3	Moderately resistant

5.4.1 Configuration Management (ACM)

5.4.1.1 Partial CM automation (ACM_AUT.1)

Developer action elements:

ACM_AUT.1.1D The developer shall use a CM system.

ACM_AUT.1.2D The developer shall provide a CM plan.

Content and presentation of evidence elements:

ACM_AUT.1.1C The CM system shall provide an automated means by which only authorized changes are made to the TOE implementation representation.

ACM_AUT.1.2C The CM system shall provide an automated means to support the generation of the TOE.

ACM_AUT.1.3C The CM plan shall describe the automated tools used in the CM system.

ACM_AUT.1.4C The CM plan shall describe how the automated tools are used in the CM system.

Evaluator action elements:

ACM_AUT.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.4.1.2 Generation support and acceptance procedures (ACM_CAP.4)

Developer action elements:

ACM_CAP.4.1D The developer shall provide a reference for the TOE.

ACM_CAP.4.2D The developer shall use a CM system.

ACM_CAP.4.3D The developer shall provide CM documentation.

Content and presentation of evidence elements:

ACM_CAP.4.1C The reference for the TOE shall be unique to each version of the TOE.

ACM_CAP.4.2C The TOE shall be labeled with its reference.

ACM_CAP.4.3C The CM documentation shall include a configuration list, a CM plan, and an acceptance plan.

ACM_CAP.4.4C The configuration list shall describe the configuration items that comprise the TOE.

ACM_CAP.4.5C The CM documentation shall describe the method used to uniquely identify the configuration items.

ACM_CAP.4.6C The CM system shall uniquely identify all configuration items.

ACM_CAP.4.7C The CM plan shall describe how the CM system is used.

ACM_CAP.4.8C The evidence shall demonstrate that the CM system is operating in accordance with the CM plan.

ACM_CAP.4.9C The CM documentation shall provide evidence that all configuration items have been and are being effectively maintained under the CM system.

ACM_CAP.4.10C The CM system shall provide measures such that only authorized changes are made to the configuration items.

ACM_CAP.4.11C The CM system shall support the generation of the TOE.

ACM_CAP.4.12C The acceptance plan shall describe the procedures used to accept modified or newly created configuration items as part of the TOE.

Evaluator action elements:

ACM_CAP.4.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.4.1.3 Problem tracking CM coverage (ACM_SCP.2)

Developer action elements:

ACM_SCP.2.1D The developer shall provide CM documentation.

Content and presentation of evidence elements:

ACM_SCP.2.1C The CM documentation shall show that the CM system, as a minimum, tracks the following: the TOE implementation representation, design documentation, test documentation, user documentation, administrator documentation, CM documentation, and security flaws.

ACM_SCP.2.2C The CM documentation shall describe how configuration items are tracked by the CM system.

Evaluator actions elements:

ACM_SCP.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.4.2 Delivery and Operation (ADO)

5.4.2.1 Detection of modification (ADO_DEL.2)

Developer action elements:

ADO_DEL.2.1D The developer shall document procedures for delivery of the TOE or parts of it to the user.

ADO_DEL.2.2D The developer shall use the delivery procedures.

Content and presentation of evidence elements:

ADO_DEL.2.1C The delivery documentation shall describe all procedures that are necessary to maintain security when distributing versions of the TOE to a user's site.

ADO_DEL.2.2C The delivery documentation shall describe how the various procedures and technical measures provide for the detection of modifications, or any discrepancy between the developer's master copy and the version received at the user site.

ADO_DEL.2.3C The delivery documentation shall describe how the various procedures allow detection of attempts to masquerade as the developer, even in cases in which the developer has sent nothing to the user's site.

Evaluator action elements:

ADO_DEL.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.4.2.2 Installation, generation, and start-up procedures (ADO_IGS.1)

Developer action elements:

ADO_IGS.1.1D The developer shall document procedures necessary for the secure installation, generation, and start-up of the TOE.

Content and presentation of evidence elements:

ADO_IGS.1.1C The documentation shall describe the steps necessary for secure installation, generation, and start-up of the TOE.

Evaluator action elements:

ADO_IGS.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADO_IGS.1.2E The evaluator shall determine that the installation, generation, and start-up procedures result in a security configuration.

5.4.3 Development (ADV)

5.4.3.1 Architectural design with justification (ADV_ARC_(EXP).1)

Developer action elements:

ADV_ARC_(EXP).1.1D The developer shall provide the architectural design of the TSF.

Content and presentation of evidence elements:

ADV_ARC_(EXP).1.1C The presentation of the architectural design of the TSF shall be informal.

ADV_ARC_(EXP).1.2C The architectural design shall be internally consistent.

ADV_ARC_(EXP).1.3C The architectural design shall describe the design of the TSF self-protection mechanisms.

ADV_ARC_(EXP).1.4C The architectural design shall describe the design of the TSF in detail sufficient to determine that the security enforcing mechanisms cannot be bypassed.

ADV_ARC_(EXP).1.5C The architectural design shall justify that the design of the TSF achieves the self-protection function.

Evaluator action elements:

ADV_ARC_(EXP).1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_ARC_(EXP).1.2E The evaluator shall analyze the architectural design and dependent documentation to determine that FPT_SEP and FPT_RVM are accurately implemented in the TSF.

5.4.3.2 Functional specification with complete summary (ADV_FSP_(EXP).1)

Developer action elements:

ADV_FSP_(EXP).1.1D The developer shall provide a functional specification.

Content and presentation of evidence elements:

ADV_FSP_(EXP).1.1C The functional specification shall completely represent the TSF.

ADV_FSP_(EXP).1.2C The functional specification shall be internally consistent.

ADV_FSP_(EXP).1.3C The functional specification shall describe the external TSF interfaces (TSFIs) using an informal style.

ADV_FSP_(EXP).1.4C The functional specification shall designate each external TSFI as security enforcing or security supporting.

ADV_FSP_(EXP).1.5C The functional specification shall describe the purpose and method of use for each external TSFI.

ADV_FSP_(EXP).1.6C The functional specification shall identify and describe all parameters associated with each external TSFI.

ADV_FSP_(EXP).1.7C For security enforcing external TSFIs, the functional specification shall describe the security enforcing effects and security enforcing exceptions.

ADV_FSP_(EXP).1.8C For security enforcing external TSFIs, the functional specification shall describe direct error messages resulting from security enforcing effects and exceptions.

Evaluator action elements:

ADV_FSP_(EXP).1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_FSP_(EXP).1.2E The evaluator shall determine that the functional specification is an accurate and complete instantiation of the user-visible TOE security functional requirements.

129 *Application Note: This requirement can potentially be met by a combination of documents provided by the developer, including the Security Target and external interface specification.*

5.4.3.3 Security-enforcing high-level design (ADV_HLD_(EXP).1)

Developer action elements:

ADV_HLD_(EXP).1.1D The developer shall provide the high-level design of the TOE.

Content and presentation of evidence elements:

ADV_HLD_(EXP).1.1C The high-level design shall describe the structure of the TOE in terms of subsystems.

ADV_HLD_(EXP).1.2C The high-level design shall be internally consistent.

ADV_HLD_(EXP).1.3C The high level design shall describe the subsystems using an informal style.

ADV_HLD_(EXP).1.4C The high-level design shall describe the design of the TOE in sufficient detail to determine what subsystems of the TOE are part of the TSF.

ADV_HLD_(EXP).1.5C The high-level design shall identify all subsystems in the TSF, and designate them as either security enforcing or security supporting.

ADV_HLD_(EXP).1.6C The high-level design shall describe the structure of the security-enforcing subsystems.

ADV_HLD_(EXP).1.7C For security-enforcing subsystems, the high-level design shall describe the design of the security-enforcing behavior.

ADV_HLD_(EXP).1.8C For security-enforcing subsystems, the high-level design shall summarize any non-security-enforcing behavior.

ADV_HLD_(EXP).1.9C The high-level design shall summarize the behavior for security-supporting subsystems.

ADV_HLD_(EXP).1.10C The high-level design shall summarize all other interactions between subsystems of the TSF.

ADV_HLD_(EXP).1.11C The high-level design shall describe any interactions between the security-enforcing subsystems of the TSF.

Evaluator action elements:

ADV_HLD_(EXP).1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_HLD_(EXP).1.2E The evaluator shall determine that the high-level design is an accurate and complete instantiation of all user-visible TOE security functional requirements with the exception of FPT_SEP and FPT_RVM.

5.4.3.4 Implementation of the TSF (ADV_IMP.2)

Developer action elements:

ADV_IMP.2.1D The developer shall provide the implementation representation for the entire TSF.

Content and presentation of evidence elements:

ADV_IMP.2.1C The implementation representation shall unambiguously define the TSF to a level of detail such that the TSF can be generated without further design decisions.

ADV_IMP.2.2C The implementation representation shall be internally consistent.

ADV_IMP.2.3C The implementation representation shall describe the relationships between all portions of the implementation.

Evaluator action elements:

ADV_IMP.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_IMP.2.2E The evaluator shall determine that the implementation representation is an accurate and complete instantiation of the TOE security functional requirements.

5.4.3.5 Modular decomposition (ADV_INT_(EXP).1)

Developer action elements:

ADV_INT_(EXP).1.1D The developer shall design and implement the TSF using modular decomposition.

ADV_INT_(EXP).1.2D The developer shall use sound software engineering principles to achieve the modular decomposition of the TSF.

ADV_INT_(EXP).1.3D The developer shall design the modules such that they exhibit good internal structure and are not overly complex.

ADV_INT_(EXP).1.4D The developer shall design modules that implement the [FDP_IFC.2(1), FDP_IFC.2(2), FDP_IFF.1-NIAP-0407(1), and FDP_IFF.1-NIAP-0407(2) requirements] such that they exhibit only functional, sequential, communicational, or temporal cohesion, with limited exceptions.

ADV_INT_(EXP).1.5D The developer shall design the SFP-enforcing modules such that they exhibit only call or common coupling, with limited exceptions.

ADV_INT_(EXP).1.6D The developer shall implement TSF modules using coding standards that result in good internal structure that is not overly complex.

ADV_INT_(EXP).1.7D The developer shall provide a software architectural description.

Content and presentation of evidence elements:

ADV_INT_(EXP).1.1C The software architectural description shall identify the SFP-enforcing and non-SFP-enforcing modules.

ADV_INT_(EXP).1.2C The TSF modules shall be identical to those described by the low level design (ADV_LLD_(EXP).1.4C).

ADV_INT_(EXP).1.3C The software architectural description shall provide a justification for the designation of non-SFP-enforcing modules that interact with the SFP-enforcing module(s).

ADV_INT_(EXP).1.4C The software architectural description shall describe the process used for modular decomposition.

ADV_INT_(EXP).1.5C The software architectural description shall describe how the TSF design is a reflection of the modular decomposition process.

ADV_INT_(EXP).1.6C The software architectural description shall include the coding standards used in the development of the TSF.

ADV_INT_(EXP).1.7C The software architectural description shall provide a justification, on a per module basis, of any deviations from the coding standards.

ADV_INT_(EXP).1.8C The software architectural description shall include a coupling analysis that describes intermodule coupling for the SFP-enforcing modules.

ADV_INT_(EXP).1.9C The software architectural description shall provide a justification, on a per module basis, for any coupling or cohesion exhibited by SFP-enforcing modules, other than those permitted.

ADV_INT_(EXP).1.10C The software architectural description shall provide a justification, on a per module basis, that the SFP-enforcing modules are not overly complex.

Evaluator action elements:

ADV_INT_(EXP).1.1E The evaluator shall confirm that the information provided meets all the requirements for content and presentation of evidence.

ADV_INT_(EXP).1.2E The evaluator shall perform a cohesion analysis for the modules that substantiates the type of cohesion claimed for a subset of SFP-enforcing modules.

ADV_INT_(EXP).1.3E The evaluator shall perform a complexity analysis for a subset of TSF modules.

5.4.3.6 Security-enforcing low-level design (ADV_LLD_(EXP).1)

Developer action elements:

ADV_LLD_(EXP).1.1D The developer shall provide the low-level design of the TSF.

Content and presentation of evidence elements:

ADV_LLD_(EXP).1.1C The presentation of the low-level design shall be informal.

ADV_LLD_(EXP).1.2C The presentation of the low-level design shall be separate from the implementation representation.

ADV_LLD_(EXP).1.3C The low-level design shall be internally consistent.

ADV_LLD_(EXP).1.4C The low-level design shall identify and describe data that are common to more than one module, where any of the modules is a security-enforcing module.

ADV_LLD_(EXP).1.5C The low-level design shall describe the TSF in terms of modules, designating each module as either security-enforcing or security-supporting.

ADV_LLD_(EXP).1.6C The low level design shall describe each security-enforcing module in terms of its purpose, interfaces, return values from those interfaces, called interfaces to other modules, and global variables.

ADV_LLD_(EXP).1.7C For each security-enforcing module, the low level design shall provide an algorithmic description detailed enough to represent the TSF implementation.

130 *Application Note: An algorithmic description contains sufficient detail such that two different programmers would produce functionally-equivalent code, although data structures, programming methods, etc. may differ.*

ADV_LLD_(EXP).1.8C The low level design shall describe each security-supporting module in terms of its purpose and interaction with other modules.

Evaluator action elements:

ADV_LLD_(EXP).1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_LLD_(EXP).1.2E The evaluator shall determine that the low-level design is an accurate and complete instantiation of all TOE security functional requirements, with the exception of FPT_SEP and FPT_RVM.

5.4.3.7 Informal correspondence demonstration (ADV_RCR.1)

Developer action elements:

ADV_RCR.1.1D The developer shall provide an analysis of correspondence between all adjacent pairs of TSF representations that are provided.

Content and presentation of evidence elements:

ADV_RCR.1.1C For each adjacent pair of provided TSF representations, the analysis shall demonstrate that all relevant security functionality of the more abstract TSF representation is correctly and completely refined in the less abstract TSF representation.

Evaluator action elements:

ADV_RCR.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.4.3.8 Informal TOE security policy model (ADV_SPM.1)

Developer action elements:

ADV_SPM.1.1D The developer shall provide a TSP model.

ADV_SPM.1.2D The developer shall demonstrate correspondence between the functional specification and the TSP model.

Content and presentation of evidence elements:

ADV_SPM.1.1C The TSP model shall be informal.

ADV_SPM.1.2C The TSP model shall describe the rules and characteristics of all policies of the TSP that can be modeled.

ADV_SPM.1.3C The TSP model shall include a rationale that demonstrates that it is consistent and complete with respect to all policies of the TSP that can be modeled.

ADV_SPM.1.4C The demonstration of correspondence between the TSP model and the functional specification shall show that all of the security functions in the functional specification are consistent and complete with respect to the TSP model.

Evaluator action elements:

ADV_SPM.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.4.4 Guidance Documents (AGD)

5.4.4.1 Administrator guidance (AGD_ADM.1)

Developer action elements:

AGD_ADM.1.1D The developer shall provide administrator guidance addressed to system administrative personnel.

Content and presentation of evidence elements:

AGD_ADM.1.1C The administrator guidance shall describe the administrative functions and interfaces available to the administer of the TOE.

AGD_ADM.1.2C The administrator guidance shall describe how to administer the TOE in a secure manner.

AGD_ADM.1.3C The administrator guidance shall contain warnings about functions and privileges that should be controlled in a secure processing environment.

AGD_ADM.1.4C The administrator guidance shall describe all assumptions regarding user behavior that are relevant to secure operation of the TOE.

AGD_ADM.1.5C The administrator guidance shall describe all security parameters under the control of the administrator, indicating secure values as appropriate.

AGD_ADM.1.6C The administrator guidance shall describe each type of security-relevant event relative to the administrative functions that need to be performed, including changing the security characteristics of entities under the control of the TSF.

AGD_ADM.1.7C The administrator guidance shall be consistent with all other documentation supplied for evaluation.

AGD_ADM.1.8C The administrator guidance shall describe all security requirements for the IT environment that are relevant to the administrator.

Evaluator action elements:

AGD_ADM.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.4.4.2 User Guidance (AGD_USR.1)

Developer action elements:

AGD_USR.1.1D The developer shall provide user guidance.

Content and presentation of evidence elements:

AGD_USR.1.1C The user guidance shall describe the functions and interfaces available to the non-administrative users of the TOE.

AGD_USR.1.2C The user guidance shall describe the use of user-accessible security functions provided by the TOE.

AGD_USR.1.3C The user guidance shall contain warnings about user-accessible functions and privileges that should be controlled in a secure processing environment.

AGD_USR.1.4C The user guidance shall clearly present all user responsibilities necessary for secure operation of the TOE, including those related to assumptions regarding user behavior found in the statement of TOE security environment.

AGD_USR.1.5C The user guidance shall be consistent with all other documentation supplied for evaluation.

AGD_USR.1.6C The user guidance shall describe all security requirements for the IT environment that are relevant to the user.

Evaluator action elements:

AGD_USR.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.4.5 Life Cycle Support (ALC)

5.4.5.1 Identification of security measures (ALC_DVS.1)

Developer action elements:

ALC_DVS.1.1D The developer shall produce development security documentation.

Content and presentation of evidence elements:

ALC_DVS.1.1C The development security documentation shall describe all the physical, procedural, personnel, and other security measures that are necessary to protect the confidentiality and integrity of the TOE design and implementation in its development environment.

ALC_DVS.1.2C The development security documentation shall provide evidence that these security measures are followed during the development and maintenance of the TOE.

Evaluator action elements:

ALC_DVS.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.4.5.2 Flaw reporting procedures (ALC_FLR.2)

Developer action elements:

ALC_FLR.2.1D The developer shall document the flaw remediation procedures.

ALC_FLR.2.2D The developer shall establish a procedure for accepting and acting upon user reports of security flaws and requests for corrections to those flaws.

Content and presentation of evidence elements:

ALC_FLR.2.1C The flaw remediation procedures documentation shall describe the procedures used to track all reported security flaws in each release of the TOE.

ALC_FLR.2.2C The flaw remediation procedures shall require that a description of the nature and effect of each security flaw be provided, as well as the stats of finding a correction to that flaw.

ALC_FLR.2.3C The flaw remediation procedures shall require that corrective actions be identified for each of the security flaws.

ALC_FLR.2.4C The flaw remediation procedures documentation shall describe the methods used to provide flaw information, corrections and guidance on corrective actions to TOE users.

ALC_FLR.2.5C The procedures for processing reported security flaws shall ensure that any reported flaws are corrected and the correction issued to TOE users.

ALC_FLR.2.6C The procedures for processing reported security flaws shall provide safeguards that any corrections to these security flaws do not introduce any new flaws.

Evaluator action elements:

ALC_FLR.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.4.5.3 Developer defined life-cycle model (ALC_LCD.1)

Developer action elements:

ALC_LCD.1.1D The developer shall establish a life-cycle model to be used in the development and maintenance of the TOE.

ALC_LCD.1.2D The developer shall provide life-cycle definition documentation.

Content and presentation of evidence elements:

ALC_LCD.1.1C The life-cycle definition documentation shall describe the model used to develop and maintain the TOE.

ALC_LCD.1.2C The life-cycle model shall provide for the necessary control over the development and maintenance of the TOE.

Evaluator action elements:

ALC_LCD.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.4.5.4 Well-defined development tools (ALC_TAT.1)

Developer action elements:

ALC_TAT.1.1D The developer shall identify the development tools being used for the TOE.

ALC_TAT.1.2D The developer shall document the selected implementation-dependent options of the development tools.

Content and presentation of evidence elements:

ALC_TAT.1.1C All development tools used for implementation shall be well-defined.

ALC_TAT.1.2C The documentation of the development tools shall unambiguously define the meaning of all statements used in the implementation.

ALC_TAT.1.3C The documentation of the development tools shall unambiguously define the meaning of all implementation-dependent options.

Evaluator action elements:

ALC_TAT.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.4.6 Tests (ATE)

5.4.6.1 Analysis Coverage (ATE_COV.2)

Developer action elements:

ATE_COV.2.1D The developer shall provide an analysis of the test coverage.

Content and presentation of evidence elements:

ATE_COV.2.1C The analysis of the test coverage shall demonstrate the correspondence between the tests identified in the test documentation and the TSF as described in the functional specification.

ATE_COV.2.2C The analysis of the test coverage shall demonstrate that the correspondence between the TSF as described in the functional specification and the tests identified in the test documentation is complete.

Evaluator action elements:

ATE_COV.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.4.6.2 Testing: low-level design (ATE_DPT.2)

Developer action elements:

ATE_DPT.1.1D The developer shall provide the analysis of the depth of testing.

Content and presentation of evidence elements:

ATE_DPT.1.1C The depth analysis shall demonstrate that the tests identified in the test documentation are sufficient to demonstrate that the TSF operates in accordance with its high-level design and low-level design.

Evaluator action elements:

ATE_DPT.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.4.6.3 Functional testing (ATE_FUN.1)

Developer action elements:

ATE_FUN.1.1D The developer shall test the TSF and document the results.

ATE_FUN.1.2D The developer shall provide test documentation.

Content and presentation of evidence elements:

ATE_FUN.1.1C The test documentation shall consist of test plans, test procedure descriptions, expected test results and actual test results.

ATE_FUN.1.2C The test plans shall identify the security functions to be tested and describe the goal of the tests to be performed.

ATE_FUN.1.3C The test procedure descriptions shall identify the tests to be performed and describe the scenarios for testing each security function. These scenarios shall include any ordering dependencies on the results of other tests.

ATE_FUN.1.4C The expected test results shall show the anticipated outputs from a successful execution of the tests.

ATE_FUN.1.5C The test results from the developer execution of the tests shall demonstrate that each tested security function behaved as specified.

Evaluator action elements:

ATE_FUN.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.4.6.4 Independent testing – sample (ATE_IND.2)

Developer action elements:

ATE_IND.2.1D The developer shall provide the TOE for testing.

Content and presentation of evidence elements:

ATE_IND.2.1C The TOE shall be suitable for testing.

ATE_IND.2.2C The developer shall provide an equivalent set of resources to those that were used in the developer’s functional testing of the TSF.

Evaluator action elements:

ATE_IND.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_IND.2.2E The evaluator shall test a subset of the TSF as appropriate to confirm that the TOE operates as specified.

ATE_IND.2.3E The evaluator shall execute a sample of tests in the test documentation to verify the developer test results.

5.4.7 Vulnerability Assessment (AVA)

5.4.7.1 Systematic cryptographic module covert channel analysis (AVA_CCA_(EXP).2)

131 *Application Note: The covert channel analysis is performed on the entire TSF to determine that TSF interfaces cannot be used covertly to obtain critical security parameters; a search is made for the leakage of critical security parameters, rather than a violation of an information control policy.*

Developer action elements:

AVA_CCA_(EXP).2.1D The developer shall conduct a search for covert channels for the leakage of critical security parameters.

AVA_CCA_(EXP).2.2D The developer shall provide covert channel analysis documentation.

Content and presentation of evidence elements:

AVA_CCA_(EXP).2.1C The analysis documentation shall identify covert channels that leak critical security parameters and estimate their capacity.

AVA_CCA_(EXP).2.2C The analysis documentation shall describe the procedures used for determining the existence of covert channels that leak critical security parameters, and the information needed to carry out the covert channel analysis.

AVA_CCA_(EXP).2.3C The analysis documentation shall describe all assumptions made during the covert channel analysis.

AVA_CCA_(EXP).2.4C The analysis documentation shall describe the method used for estimating channel capacity, based on worst-case scenarios.

AVA_CCA_(EXP).2.5C The analysis documentation shall describe the worst-case exploitation scenario for each identified covert channel.

AVA_CCA_(EXP).2.6C The analysis documentation shall provide evidence that the method used to identify covert channels is systematic.

Evaluator action elements:

AVA_CCA_(EXP).2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_CCA_(EXP).2.3E The evaluator shall selectively validate the covert channel analysis through independent analysis and testing.

132 *Application Note: The cryptographic security parameters are defined in FIPS 140-2.*

5.4.7.2 Validation of analysis (AVA_MSU.2)

Developer action elements:

AVA_MSU.2.1D The developer shall provide guidance documentation.

AVA_MSU.2.2D The developer shall document an analysis of the guidance documentation.

Content and presentation of evidence elements:

AVA_MSU.2.1C The guidance documentation shall identify all possible modes of operation of the TOE (including operation following failure or operational error), their consequences and implications for maintaining secure operation.

AVA_MSU.2.2C The guidance documentation shall be complete, clear, consistent and reasonable.

AVA_MSU.2.3C The guidance documentation shall list all assumptions about the intended environment.

AVA_MSU.2.4C The guidance documentation shall list all requirements for external security measures (including external procedural, physical and personnel controls).

AVA_MSU.2.5C The analysis documentation shall demonstrate that the guidance documentation is complete.

Evaluator action elements:

AVA_MSU.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_MSU.2.2E The evaluator shall repeat all configuration and installation procedures, and other procedures selectively, to confirm that the TOE can be configured and used securely using only the supplied guidance documentation.

AVA_MSU.2.3E The evaluator shall determine that the use of the guidance documentation allows all insecure states to be detected.

AVA_MSU.2.4E The evaluator shall confirm that the analysis documentation shows that guidance is provided for secure operation in all modes of operation of the TOE.

5.4.7.3 Strength of TOE security function evaluation (AVA_SOF.1)

Developer action elements:

AVA_SOF.1.1D The developer shall perform a strength of TOE security function analysis for each mechanism identified in the ST as having a strength of TOE security function claim.

Content and presentation of evidence elements:

AVA_SOF.1.1C For each mechanism with a strength of TOE security function claim the strength of TOE security function analysis shall show that it meets or exceeds the minimum strength level of SOF-basic.

AVA_SOF.1.2C For each mechanism with a specific strength of TOE security function claim the strength of TOE security function analysis shall show that it meets or exceeds the specific strength of function metric of SOF-basic.

Evaluator action elements:

AVA_SOF.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_SOF.1.2E The evaluator shall confirm that the strength claims are correct.

5.4.7.4 Moderately resistant (AVA_VLA.3)

Developer action elements:

AVA_VLA.3.1D The developer shall perform and document an analysis of the TOE deliverables searching for ways in which a user can violate the TSP.

AVA_VLA.3.2D The developer shall document the disposition of identified vulnerabilities.

Content and presentation of evidence elements:

AVA_VLA.3.1C The documentation shall show, for all identified vulnerabilities, that the vulnerability cannot be exploited in the intended environment for the TOE.

AVA_VLA.3.2C The documentation shall justify that the TOE, with the identified vulnerabilities, is resistant to obvious penetration attacks.

AVA_VLA.3.3C The evidence shall show that the search for vulnerabilities is systematic.

Evaluator action elements:

AVA_VLA.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_VLA.3.2E The evaluator shall conduct penetration testing, building on the developer vulnerability analysis, to ensure the identified vulnerabilities have been addressed.

AVA_VLA.3.3E The evaluator shall perform an independent vulnerability analysis.

AVA_VLA.3.4E The evaluator shall perform independent penetration testing, based on the independent vulnerability analysis, to determine the exploitability of additional identified vulnerabilities in the intended environment.

AVA_VLA.3.5E The evaluator shall determine that the TOE is resistant to penetration attacks performed by an attacker possessing a moderate attack potential.

6 RATIONALE

133 This section provides the rationale for the selection of the IT security requirements, objectives, assumptions, and threats. In particular, it shows that the IT security requirements are suitable to meet the security objectives, which in turn are shown to be suitable to cover all aspects of the TOE security environment.

6.1 Rationale for TOE Security Objectives

Table 14 Rationale for TOE Security Objectives

Threat/Policy	Objectives	Rationale
T.ADMIN_ERROR An administrator may incorrectly install or configure the TOE, or install a corrupted TOE resulting in ineffective security mechanisms.	O.ADMIN_GUIDANCE The TOE will provide administrators with the necessary information for secure delivery and management.	O.ADMIN_GUIDANCE (ADO_DEL.2, ADO_IGS.1, AGD_ADM.1, AGD_USR.1, AVA_MSU.2) help to mitigate this threat by ensuring the TOE administrators have guidance that instructs them how to administer the TOE in a secure manner and to provide the administrator with instructions to ensure the TOE was not corrupted during the delivery process.
	O.ADMIN_ROLE The TOE will provide an administrator role to isolate administrative actions.	O.ADMIN_ROLE (FMT_SMR.2, FMT_SMR.3) plays a role in mitigating this threat by limiting the functions an administrator can perform. The authorized user must explicitly request to transfer into an administrative role. This ensures they cannot change configuration settings accidentally during their legitimate use of the PED.

Threat/Policy	Objectives	Rationale
	O.MANAGE The TOE will provide all the functions and facilities necessary to support the administrator in their management of the security of the TOE, and restrict these functions and facilities from unauthorized use.	O.MANAGE FMT_MTD.1 also contributes to mitigating this threat by providing administrators the capability to view configuration settings. For example, if the Administrator made a mistake when configuring the rule-set, providing them the capability to view the rules affords them the ability to review the rules and discover any mistakes that might have been made.
T.AUDIT_COMPROMISE A malicious user or process may view audit records, cause audit records to be lost or modified, or prevent future audit records from being recorded, thus masking a users action	O.AUDIT_PROTECTION	O.AUDIT_PROTECTION FAU.SAR.2, FAU_STG.1-NIAP-0429, FAU_STG.3, FAU_STG.NIAP-0414, FMT_SMF.1) contributes to mitigating this threat by controlling access to the audit trail. The auditor and any trusted IT entities performing IDS-like function are the only ones allowed to read the audit trai. No one is allowed to modify audit records, and the Auditor is the only allowed to delete audit records in the audit trail. The TOE has the capability to prevent auditable actions from occurring if the audit trail is full, and of notifying an administrator if the audit trail is approaching its capacity. In addition, the TOE has the capability to restore audit data corrupted by the attacker.

Threat/Policy	Objectives	Rationale
	O.RESIDUAL_INFORMATION	O.RESIDUAL_INFORMATION (FDP_RIP.2) prevents a user not authorized to read the audit trail from access to audit information that might otherwise be persistent in a TOE resource (e.g. memory). By ensuring the TOE prevents residual information in a resource, audit information will not become available to any user or process except those explicitly authorized for that data.
	O.SELF_PROTECTION The TSF will maintain a domain for its own execution that protects itself and its resources from external interference, tampering or unauthorized disclosure	O.SELF_PROTECTION (FPT_SEP.2, FPT_RVM.1) contributes to countering this threat by ensuring that the TSF can protect itself from users. If the TSF could not maintain and control its domain of execution, it could not be trusted to control access to the resources under its control, which includes the audit trail. Likewise, ensuring that the functions that protect the audit trail are always invoked is also critical to the mitigation of this threat.

Threat/Policy	Objectives	Rationale
T.CRYPTO_COMPROMISE A malicious user or process may cause key, data or executable code associated with the cryptographic functionality to be inappropriately accessed (viewed, modified, or deleted), thus compromising the cryptographic mechanisms and the data protected by those mechanisms.	O.DOCUMENT_KEY_LEAKAGE The bandwidth of channels that can be used to compromise key material shall be documented.	O.DOCUMENT_KEY_LEAKAGE (AVA_CCA_(EXP).2) addresses this threat by requiring the developer to perform an analysis that documents the amount of key information that can be leaked via a covert channel. This provides information that identifies how much material could be inappropriately obtained within a specified time period.
	O.SELF_PROTECTION The TSF will maintain a domain for its own execution that protects itself and its resources from external interference, tampering, or unauthorized disclosure.	O.SELF_PROTECTION (FPT_SEP.2, FPT_RVM.1) contributes to countering this threat by ensuring that the TSF can protect itself from users. If the TSF could not maintain and control its domain of execution, it could not be trusted to control access to the resources under its control, which includes the cryptographic data and executable code.

Threat/Policy	Objectives	Rationale
T.FLAWED_DESIGN Unintentional or intentional errors in requirements specification or design of the TOE may occur, leading to flaws that may be exploited by a malicious user or program.	O.CHANGE_MANAGEMENT The configuration of, and all changes to, the TOE and its development evidence will be analyzed, tracked, and controlled throughout the TOE's development.	O.CHANGE_MANAGEMENT (ACM_AUT.1, ACM_CAP.4, ACM_SCP.2, ALC_DVS.1, ALC_FLR.2, ALC_LCD.1) plays a role in countering this threat by requiring the developer to provide control of the changes made to the TOE's design. This includes controlling physical access to the TOE's development area, and having an automated configuration management system that ensures changes made to the TOE go through an approval process and only those persons that are authorized can make changes to the TOE's design and its documentation.
	O.SOUND_DESIGN The TOE will be designed using sound design principles and techniques. The TOE design, design principles and design techniques will be adequately and accurately documented.	O.SOUND_DESIGN (ADV_FSP_(EXP).1, ADV_HLD_(EXP).1, ADV_INT_(EXP).1, ADV_LLD_(EXP).1, ADV_ARC_(EXP).1, ADV_RCR.1, ADV_SPM.1) counters this threat, to a degree, by requiring that the TOE be developed using sound engineering principles. By accurately and completely documenting the design of the security mechanisms in the TOE, including a security model, the design of the TOE can be better understood, which increases the chances that design errors will be discovered.

Threat/Policy	Objectives	Rationale
	O.VULNERABILITY_ANALYSIS_TEST The TOE will undergo appropriate independent vulnerability analysis and penetration testing to demonstrate the design and implementation of the TOE does not allow attackers with medium attack potential to violate the TOE's security policies.	O.VULNERABILITY_ANALYSIS_TEST (AVA_VLA.3) ensures that the design of the TOE is independently analyzed for design flaws. Having an independent party perform the assessment ensures an objective approach is taken and may find errors in the design that would be left undiscovered by developers that have a preconceived incorrect understanding of the TOE's design.
T.FLAWED_IMPLEMENTATION Unintentional or intentional errors in implementation of the TOE design may occur, leading to flaws that may be exploited by a malicious user or program.	O.CHANGE_MANAGEMENT The configuration of, and all changes to, the TOE and its development evidence will be analyzed, tracked, and controlled throughout the TOE's development.	O.CHANGE_MANAGEMENT (ACM_CAP.4, ACM_SCP.2, ALC_DVS.1, ALC_FLR.2, ALC_LCD.1, ACM_AUT.1) This objective plays a role in mitigating this threat in the same way that the flawed design threat is mitigated. By controlling who has access to the TOE's implementation representation and ensuring that changes to the implementation are analyzed and made in a controlled manner, the threat of intentional or unintentional errors being introduced into the implementation are reduced.

Threat/Policy	Objectives	Rationale
	O.SOUND_IMPLEMENTATION The implementation of the TOE will be an accurate instantiation of its design, and is adequately and accurately documented.	In addition to documenting the design so that implementers have a thorough understanding of the design, O.SOUND_IMPLEMENTATION (ADV_IMP.2, ADV_LLD_(EXP).1, ADV_RCR.1, ADV_INT_(EXP).1, ADV_ARC_(EXP).1, ALC_TAT.1) requires that the developer's tools and techniques for implementing the design are documented. Having accurate and complete documentation, and having the appropriate tools and procedures in the development process helps reduce the likelihood of unintentional errors being introduced into the implementation.
	O.THOROUGH_FUNCTIONAL_TESTING The TOE will undergo appropriate security functional testing that demonstrates the TSF satisfies the security functional requirements.	Although the previous three objectives help minimize the introduction of errors into the implementation, O.THOROUGH_FUNCTIONAL_TESTING (ATE_COV.2, ATE_FUN.1, ATE_DPT.2, ATE_IND.2) increases the likelihood that any errors that do exist in the implementation (with respect to the functional specification, high level, and low-level design) will be discovered through testing.

Threat/Policy	Objectives	Rationale
	O.VULNERABILITY_ ANALYSIS_TEST The TOE will undergo appropriate independent vulnerability analysis and penetration testing to demonstrate the design and implementation of the TOE does not allow attackers with medium attack potential to violate the TOE's security policies.	O.VULNERABILITY_ANALYSIS_TEST (AVA_VLA.3) helps reduce errors in the implementation that may not be discovered during functional testing. Ambiguous design documentation and the fact that exhaustive testing of the external interfaces is not required may leave bugs in the implementation undiscovered in functional testing. Having an independent party perform a vulnerability analysis and conduct testing outside the scope of functional testing increases the likelihood of finding errors.
T.LOSS_OR_THEFT Portable devices might be lost or stolen allowing a malicious user to use that device to gain sensitive information	O.ROBUST_TOE_ACCESS The TOE will provide mechanisms that control a user's logical access to the TOE and to explicitly deny access to specific users when appropriate.	O.ROBUST_TOE_ACCESS (FIA_AFL.1, FIA_UAU.2, FIA_UID.2, FIA_USB.1-NIAP-0415) mitigates this threat by requiring the TOE to identify and authenticate the authorized user prior to allowing any TOE access or any TOE mediated access on behalf of that user. Only a certain number of authentication failures will be permitted. If that threshold is surpassed, the device will be locked for a pre-determined amount of time.

Threat/Policy	Objectives	Rationale
	O.ENCRYPT_STORED_DATA All user data stored within the TOE will be encrypted using TDEA with a key size of 168 bits.	O.ENCRYPT_STORED_DATA (FCS_COP.1(1), FDP_ACF.1, FDP_ACC.2) ensures that all the user data that is stored within the TOE will be encrypted using TDEA with a key size of 168 bits. This will help to mitigate the threat of the PED being lost or stolen and the attacker opening the PED to bypass the authentication and identification mechanisms.
T.MALICIOUS_TSF_COMPROMISE A malicious user or process may cause TSF data or executable code to be inappropriately accessed (viewed, modified, or deleted).	O.SELF_PROTECTION The TSF will maintain a domain for its own execution that protects itself and its resources from external interference, tampering or unauthorized disclosure.	O.SELF_PROTECTION (FPT_SEP.2, FPT_RVM.1) requires that the TSF be able to protect itself from tampering and that the security mechanisms in the TSF cannot be bypassed. Without this objective, there could be no assurance the authorized user could not view or modify TSF data or TSF executables.
	O.MANAGE The TOE will provide all the functions and facilities necessary to support the administrator in their management of the security of the TOE, and restrict these functions and facilities from unauthorized use.	O.MANAGE (FMT_MTD.1, FMT_MOF.1(all iterations), FMT_SMF.1) provides the capability to restrict access to TSF to those roles that are authorized to use the functions. Satisfaction of this objective (and its associated requirements) prevents unauthorized access to TSF functions and data through the administrative mechanism.

Threat/Policy	Objectives	Rationale
	O.DISPLAY_BANNER The TOE will display an advisory warning regarding use of the TOE.	O.DISPLAY_BANNER (FTA_TAB.1) helps mitigate this threat by providing the Administrator the ability to remove product information (e.g., product name, version number) from the banner that is displayed to the user. Having product information about the TOE provides an attacker with information that may increase their ability to compromise the TOE.
	O.TRUSTED_PATH The TOE will provide a means to ensure that users are not communicating with some other entity pretending to be the TOE when supplying identification and authentication data.	O.TRUSTED_PATH (FTP_TRP.1) plays a role in addressing this threat by ensuring that there is a trusted communication path between the TSF and the user. This ensures the transmitted data cannot be compromised or disclosed during the duration of the trusted path. The protection offered by this objective is limited to TSF data, including authentication data.

Threat/Policy	Objectives	Rationale
T.MASQUERADE A malicious user, process, or external IT entity may masquerade as an authorized entity in order to gain access to data or TOE resources.	O.ROBUST_TOE_ACCESS The TOE will provide mechanisms that control a user's logical access to the TOE and to explicitly deny access to specific users when appropriate.	O.ROBUST_TOE_ACCESS (FIA_AFL.1, FIA_ATD.1, FIA_UID.2, FIA_UAU.2, FIA_USB.1-NIAP-0415, AVA_SOF.1) mitigates this threat by controlling the logical access to the TOE and its resources. By mandating the type and strength of the authentication mechanisms, this objective helps mitigate the possibility of a user attempting to login and masquerade as an authorized user. In addition, this objective provides the Administrator the means to control the number of failed login attempts the user can generate before an account is locked out, further reducing the possibility of a user gaining unauthorized access to the TOE.

Threat/Policy	Objectives	Rationale
T.POOR_TEST Lack of or insufficient tests to demonstrate that all TOE security functions operate correctly (including in a fielded TOE) may result in incorrect TOE behavior being undiscovered thereby causing potential security vulnerabilities.	O.CORRECT_TSF_OPERATION The TOE will provide a capability to test the TSF to ensure the correct operation of the TSF in its operational environment.	While the testing performed for O.THOROUGH_FUNCTIONAL_TESTING are necessary for successful completion of an evaluation, this testing activity does not address the concern that the TOE continues to operate correctly and enforce its security policies once it has been fielded. Some level of testing must be available to end users to ensure the TOE's security mechanisms continue to operate correctly once the TOE is fielded. O.CORRECT_TSF_OPERATION (FPT_TST_(EXP).4, FPT_TST_(EXP).5) ensures that once the TOE is distributed to an end user, the capability exists that the integrity of the TSF (hardware and software, including the cryptographic functions) can be demonstrated, and thus providing end users the confidence that the TOE's security policies continue to be enforced.

Threat/Policy	Objectives	Rationale
	O.THOROUGH_FUNCTIONAL_TESTING The TOE will undergo appropriate security functional testing that demonstrates the TSF satisfies the security functional requirements.	Design analysis determines that TOE's documented design satis fies the security functional requirements. In order to ensure the TOE's deign is correctly realized in its implementation, the appropriate level of functional testing of the TOE's security mechanisms must be performed during the evaluation of the TOE. O.THOROUGH_FUNCTIONAL_TESTING (ATE_FUN.1, ATE_COV.2, ATE_DPT.2, ATE_IND.2) ensures that adequate functional testing is performed to demonstrate the TSF satisfies the security functional requirements and that the TOE's security mechanisms operate as documented. While functional testing serves an important purpose, it does not ensure the TSFI cannot be used in unintended ways to circumvent the TOE's security policies.

Threat/Policy	Objectives	Rationale
	O.VULNERABILITY_ ANALYSIS_TEST The TOE will undergo appropriate independent vulnerability analysis and penetration testing to demonstrate the design and implementation of the TOE does not allow attackers with medium attack potential to violate the TOE's security policies.	O.VULNERABILITY_ANALYSIS_TEST (AVA_VLA.3) addresses this concern by requiring a vulnerability analysis be performed in conjunction with testing that goes beyond functional testing. This objective provides a measure of confidence that the TOE does not contain security flaws that may not be identified through functional testing.
T.REPLAY A malicious user or process may capture and replay encrypted transmissions, thus assuming another senders identity.	O.REPLAY The TOE will provide a means to detect and reject the replay of authentication data, as well as, TSF data and security attributes.	O.REPLAY FPT_RPL.1 prevents a user from replaying encrypted transmissions that might have been captured as they were transmitted.
T.RESIDUAL_DATA A user or process may gain unauthorized access to data through reallocation of TOE resources from one user or process to another.	O.RESIDUAL_INFORMATION The TOE will ensure that any information contained in a protected resource is not released when the resource is reallocated	O.RESIDUAL_INFORMATION FDP_RIP.2 (counters this threat by ensuring that TSF data and user data is not persistent when resources are released by one user/process and allocated to another user/process.

Threat/Policy	Objectives	Rationale
T.RESOURCE_EXHAUSTION A malicious process or user may block other from system resources via a resource exhaustion denial of service attack.	O.RESOURCE_SHARING The TOE shall provide mechanisms that mitigate attempts to exhaust resources provided by the TOE.	O.RESOURCE_SHARING (FRU_RSA.1, FMT_MTD.2) mitigates this threat by requiring the TOE to provide controls relating to two different resources: CPU time and available network connections. The administrator is allowed to specify a percentage of processor time that is allowed to be used so that an attempt to exhaust the resource will fail when it reaches the quota. This objective also addresses the denial of services attack of a user attempting to exhaust the connection-oriented resources by generating a lagre number of half open connections (e.g. SYN attack).
T.SPOOFING A malicious user, process, or external IT entity may misrepresent itself as the TOE to obtain identification and authentication data.	O.TRUSTED_PATH The TOE will provide a means to ensure that users are not communicating with some other entity pretending to be the TOE when supplying identification and authentication data.	O.TRUSTED_PATH (FTP_TRP.1) satisfies this policy by requiring that each authentication attempt is conducted via a secure channel.

Threat/Policy	Objectives	Rationale
T.UNATTENDED_SESSION A user may gain unauthorized access to an unattended session.	O.ROBUST_TOE_ACCESS The TOE will provide mechanisms that control a user's logical access to the TOE and to explicitly deny access to specific users when appropriate.	O.ROBUST_TOE_ACCESS (FTA_SSL.1, FTA_SSL.2) helps to mitigate this threat by including mechanisms that place controls on the authorized user's session. The user's session will be locked after a Administrator-defined time period of inactivity. Locking the user's session (by TSF or user initiated) reduces the opportunity of someone gaining unauthorized access to the session when the device is unattended.

Threat/Policy	Objectives	Rationale
T.UNAUTHORIZED_ACCESS A process may gain access to user data for which it is not authorized according to the TOE security policy.	O.MEDIATE The TOE must protect user data in accordance with its security policy.	O.MEDIATE (FDP_IFC.2(1), FDP_IFF.1-NIAP-0407(1)) ensures the Application Separation Policy will keep proper separation of application data so that applications may access only that data for which they are permitted by the rules of the policy. This will prevent malicious programs from tampering with user data meant for other programs. O.MEDIATE (FDP_IFC.2(2), FDP_IFF-NIAP-0407(2)) also ensures the SCIF Mode Policy will not allow any data to be emitted or collected while in a SCIF, whether it be by microphone, speaker, or some other means. O.MEDIATE (FDP_ACC.2, FDP_ACF.1) ensures that the Stored Data Policy mitigates the threat that users could gain access to user data stored on the TOE by encrypting data stored in persistent memory.

Threat/Policy	Objectives	Rationale
	O.USER_GUIDANCE The TOE will provide users with the information necessary to correctly user the security mechanisms.	O.USER_GUIDANCE (AGD_USR.1) mitigates this threat by providing the user the information necessary to user the security mechanisms that control access to user data in a secure manner. For instance, the method by which the Application Separation Policy mechanisms (FDP_IFF.1-NIAP-0407(1), FDP_IFC.2(1)) and the SCIF Mode Policy mechanisms (FDP_IFC.2(2), FDP_IFF.1-NIAP-0407) are configured, and how to apply it to the data the user owns, is described in the user guidance. If this information were not available to the user, the information may be left unprotected, or the user may mis-configure the controls and unintentionally allow unauthorized access to their data.
T.UNIDENTIFIED_ACTIONS The administrator may fail to notice potential security violations, thus limiting the administrator's ability to identify and take action agains a possible security breach.	O.AUDIT_REVIEW The TOE will provide the capability to selectively view audit information, and alert the administrator of identified potential security violations.	O.AUDIT.REVIEW (FAU_SAA.1-NIAP-0407, FAU_ARP.1, FAU_ARP_ACK_DIR(EXP).1, FAU_SAR.1, FAU_SAR.3) helps to mitigate this threat by providing a variety of mechanisms for monitoring the use of the system. The two basic ways audit review is performed is through analysis of the audit trail produced by the audit mechanism, and through the use of automated analysis

Threat/Policy	Objectives	Rationale
		and alarm system. For analyzing the audit trail, the TOE requires an Administrator Role. This role may review and delete the audit trail for maintenance. A search and sort capability provides an efficient mechanism for the administrator to view pertinent audit information. In addition, the TOE has the capability to export the audit information to an external audit analysis tool (such as an intrusion detection system) for more details or composite audit analysis. The TOEs audit analysis mechanism must consist of a minimum set of configurable audit events that could indicate a potential security violation. Thresholds for these events must be configurable by the administrator role. By configuring these auditable events the TOE monitors the occurrences of these events (e.g. set number of authentication failures, self-test failures, etc) and immediately notifies an administrator once an event has occurred or a set threshold has been met.

Threat/Policy	Objectives	Rationale
T.UNKNOWN_STATE When the TOE is initially started or restarted after a failure, the security state of the TOE may be unknown.	O.MAINT_MODE The TOE shall provide a mode from which recovery or initial startup procedures can be performed.	O.MAIN_MODE (FPT_RCV.2) helps to mitigate this threat by ensuring that the TOE does not continue to operate in an insecure state when a hardware or software failure occurs. After a failure, the TOE enters a state that disallows operations and requires an administrator to follow documented procedures to return the TOE to a secure state.
	O.CORRECT_TSF_OPERATION The TOE will provide a capability to test the TSF to ensure the correct operation of the TSF in its operation environment.	O.CORRECT_TSF_OPERATION (FPT_TST_(EXP).4, FPT_TST_(EXP).5) counters this threat by ensuring that the TSF runs a suite of tests to successfully demonstrate the correct operation of the TSF (hardware and software_ and the TSF's cryptographic components at initial startup of the TOE. In addition to ensuring that the TOE's security state can be verified, the administrative role can verify the integrity of the TSF's data and stored code as well as the TSF's cryptographic data and stored code using the TOE-provided cryptographic mechanisms.

Threat/Policy	Objectives	Rationale
P.ACCESS_BANNER The TOE shall display an initial banner describing restrictions of use, legal agreements, or any other appropriate information to which users consent by accessing the TOE.	O.DISPLAY_BANNER The TOE will display an advisory warning regarding use of the TOE.	O.DISPLAY_BANNER (FTA_TAB.1) satisfies this policy by ensuring that the TOE displays a Security Administrator-configurable banner that provides the user with a warning about the unauthorized use of the TOE before each session is established.
P.ACCOUNTABILITY The authorized user of the TOE shall be held accountable for their actions within the TOE.	O.AUDIT_GENERATION The TOE will provide the capability to detect and create records of security relevant events associated with users or processes.	O.AUDIT_GENERATION (FAU_GEN.1-NIAP-0407, FAU_GEN.2-NIAP-410, FIA_USB.1-NIAP-0415, FAU_SEL.1-NIAP-0407) addresses this policy by providing an audit mechanism to record the actions of specfifc user/process, as well as the capability for an administrator to “preselect” audit events based on the user/process ID.The audit event selection function is configurable during run-time to ensure the TOE is able to capture security-relevant events given changes in threat conditions. Attributes used in the audit record generation are also require to be bound to the subjects, ensuring users are held accountable.

Threat/Policy	Objectives	Rationale
	O.TIME_STAMPS The TOE shall provide reliable time stamps and the capability for the administrator to set the time used for these time stamps.	O.TIME_STAMPS (FPT_STM.1, FMT_MTD.1) plays a role in supporting this policy by requiring the TOE to provide a reliable time stamp (configured locally by the administrator via a trusted IT entity, such as an NTP server). The audit mechanism is required to include the current date and time in each audit record. All audit records that include the user/process ID will also include the date and time that the event occurred.
	O.ROBUST_TOE_ACCESS The TOE will provide mechanisms that control a user's logical access to the TOE and to explicitly deny accesss to specific users when appropriate.	O.ROBUST_TOE_ACCESS (FIA_UID.2, FIA_UAU.2,) supports this policy by requiring the TOE to identify and authenticate all authorized users prior to allowing any TOE access or any TOE mediated access on behalf of these users. Note that although the TSF allows access by anonymous users , this objective does not apply to such users because they are not authenticated.
P.ADMIN_ACCESS Administrators shall be able to administer the TOE locally through protected communications channels.	O.ADMIN_ROLE The TOE will provide an administrator role to isolate administrative actions.	O.ADMIN_ROLE (FMT_SMR.2, FMT_SMR.3) supports this policy by requiring the TOE to provide mechanisms for the authorized user to explicitly request to assume a administrative role and to allow local administration of the TOE only while the user is assuming an administrative role.

Threat/Policy	Objectives	Rationale
	O.TRUSTED_PATH The TOE will provide a means to ensure that users are not communicating with some other entity pretending to be the TOE when supplying identification and authentication data.	O.TRUSTED_PATH (FTP_TRP.1) satisfies this policy by requiring that each authentication attempt and request for an administrative session is conducted via a secure channel.
P.CRYPTOGRAPHIC_FUNCTIONS The TOE shall provide cryptographic functions for its own use, including encryption/decryption and digital signature operations.	O.CRYPTOGRAPHIC_FUNCTIONS The TOE shall provide cryptographic functions for its own use, including encryption/decryption and digital signature operations.	O.CRYPTOGRAPHIC_FUNCTIONS (FCS_CKM.1, FCS_CKM.2, FCS_CKM.4, FCS_CKM_(EXP).1 FCS_CKM_(EXP).2, FCS_COA_(EXP).1, FCS_COP.1(1), FCS_COP.1(2), FCS_COP_(EXP).1) Implements this policy requiring a combination of FIPS-validation and non-FIPS-validated cryptographic mechanisms that are used to provide encryption/decryption services, as well as digital signature functions. Functions include symmetric encryption and decryption, digital signatures, as well as key generation and establishment functions.

Threat/Policy	Objectives	Rationale
P.CRYPTOGRAPHY_VALIDATED Where the TOE requires FIPS-approved security functions, only NIST FIPS validated cryptography (methods and implementations) are acceptable for key management (i.e.; generation, access, distribution, destruction, handling, and storage of keys) and cryptographic services (i.e.; encryption, decryption, signature, hashing, key distribution, and random number generation services	O.CRYPTOGRAPHY_VALIDATED The TOE shall use NIST FIPS 140-2 validated cryptomodules for cryptographic services implementing FIPS-approved security functions and random number generation services used by cryptographic functions	O.CRYPTOGRAPHY_VALIDATED (FCS_BCM_(EXP).1, FCS_COP_(EXP).1)satisfies this policy by requiring the TOE to implement NIST FIPS validated cryptographic services. These services will provide confidentiality and integrity protection of TSF data.
P.ENCRYPT_STORED_DATA The TOE shall encrypt all user data that is stored within the TOE using TDEA and a key size of 168 bits.	O.ENCRYPT_STORED_DATA All user data stored within the TOE will be encrypted using TDEA with a key size of 168 bits.	O.ENCRYPT_STORED_DATA (FCS_COP.1(1), FDP_ACF.1, FDP_ACC.2) ensures that all the user data that is stored within the TOE will be encrypted using TDEA with a key size of 168 bits. This will help to mitigate the threat of the PED being lost or stolen and the attacker opening the PED to bypass the authentication and identification mechanisms.

Threat/Policy	Objectives	Rationale
P.PDA_PKI DOD class 4, Version 3 X.509 certificates shall be used as appropriate for encryption and to digitally sign wireless transmissions.	O.CRYPTOGRAPHIC_FUNCTIONS The TOE shall provide cryptographic functions for its own use, including encryption/decryption and digital signature operations.	O.CRYPTOGRAPHIC_FUNCTION (FCS_CKM.1(2), FCS_COP.1(2), FCS_COP.1(3), FCS_CKM_(EXP).1, FCS_COA_(EXP).1 provide cryptographic operations to the TOE which support the PKI. This includes digital signatures and generation of asymmetric keys.
P.SCIF_MODE The TOE must not collect or record any audio or video data or emit electronic communications while in a SCIF.	O.MEDIATE The TOE must protect user data in accordance with its security policy.	O.MEDIATE (FDP_IFC.2(2), FDP_IFF-NIAP-0407(2)) satisfies this policy by ensuring no data will flow into the TOE or out of the TOE when the device is put into SCIF Mode. For example, data will not be able to be collected through the recording conversations on the microphone. Similarly, data will not be permitted to emit through a speaker or other electronic device listed in the appendices.
P.TRANSPORT_PROTECTION The TOE shall provide encryption and signature services to protect user data while it is being transmitted to and from the TOE.	O.CRYPTOGRAPHIC_FUNCITONS The TOE shall provide cryptographic functions for its own use, including encryption/decryption and digital signature operations.	O.CRYPTOGRAPHIC_FUNCTION (FCS_CKM.1, FCS_COP.1(1)) provide the cryptographic functions that support the PDA's ability to transfer data according to its policy.

Threat/Policy	Objectives	Rationale
	O.MEDIATE The TOE must protect user data in accordance with its security policy.	O.MEDIATE (FDP_IFC.1 FDP_IFF.1-NIAP-0407(3)) ensures that data transmitted from the TOE is encrypted and signed in accordance with the External Flow Policy. Data received by the TOE will have signature verification and will be decrypted by the TOE in accordance with the policy.
P.VULNERABILITY_ANALYSIS_TEST The TOE must undergo appropriate independent vulnerability analysis and penetration testing to demonstrate that the TOE is resistant to an attacker possessing a medium attack potential.	O.VULNERABILITY_ANALYSIS_TEST The TOE will undergo appropriate independent vulnerability analysis and penetration testing to demonstrate the design and implementation of the TOE does not allow attackers with medium attack potential to violate the TOE's security policies.	O.VULNERABILITY_ANALYSIS_TEST (AVA_VLA.3) satisfies this policy by ensuring that an independent analysis is performed on the TOE and penetration testing based on that analysis is performed. Having an independent party perform the analysis helps ensure objectivity and eliminates preconceived notions of the TOE's design and implementation that may otherwise affect the thoroughness of the analysis. The level of analysis and testing requires that an attacker with a moderate attack potential cannot compromise the TOE's ability to enforce its security policies.

6.1.1 Cellular Communications Package Security Objectives Rationale

The rationale listed here for each of the Threats/Policies must be combined with the Rationale Table in the base PED PP to achieve a completed mapping. Only those Objectives with the stated rationale are listed in this table.

Table 15 Mapping of Threats to Objectives for Cellular Communications Package

Threat/Policy	Objectives Addressing the Threat/Policy	Rationale
P.TRANSPORT_PROTECTION The TOE shall provide encryption and signature services to protect user data while it is being transmitted to and from the TOE.	O.MEDIATE The TOE must protect user data in accordance with its security policy.	O.MEDIATE (FDP_IFC[CELL].1(A), FDP_IFF[CELL].1-NIAP-0407(A)) ensures that data transmitted from the TOE to a remote receiver is encrypted and signed in accordance with the Voice Transport Protection Policy. Data received by the TOE will have signature verification and will be decrypted by the TOE in accordance with the policy.
		O.MEDIATE (FDP_IFC[CELL].1(B), FDP_IFF[CELL].1-NIAP-0407(B)) ensures that data transmitted from the TOE to a remote receiver is encrypted and signed in accordance with the Push to Talk Over Cellular Transport Protection Policy. Data received by the TOE will have signature verification and will be decrypted by the TOE in accordance with the policy.

6.2 Rationale for the Security Objectives and Security Functional Requirements for the Environment

134 All of the security objectives for the environment are restatements of an assumption found in Section 3. Therefore, those security objectives for the non-IT environment trace to the assumptions trivially and are suitable for covering the assumptions.

6.3 Rationale for TOE Security Requirements

Table 16 Rationale for TOE Security Requirements

Objective	Requirements Addressing the Objective	Rationale
O.ADMIN_ROLE The TOE will provide an administrator role to isolate administrative actions.	FMT_SMR.2	FMT_SMR. The TSF is able to associate the human user with one or more roles and these roles isolate administrative functions in that the functions of these roles only overlap in that all roles can invoke self-tests.
	FMT_SMR.3	FMT_SMR.3 requires the authorized user to explicitly request entry into an administrative role. The user should only transition into an administrative role to perform administrative functions. This will reduce the chances the user will accidentally change configuration settings.
O.ADMIN_GUIDANCE The TOE will provide administrators with the necessary information for secure delivery and management.	ADO_DEL.2	ADO_DEL.2 ensures that the administrator is provided documentation that instructs them how to ensure the delivery of the TOE, in whole or in parts, has not been tampered with or corrupted during delivery. This requirement ensures that administrator has the ability to begin their TOE installation with a clean (e.g., malicious code has not been inserted once it has left the developer's control) version of the TOE, which is necessary for secure management of the TOE.

Objective	Requirements Addressing the Objective	Rationale
	ADO_IGS.1	The ADO_IGS.1 requirement ensures the administrator has the information necessary to install the TOE in the evaluated configuration. Often times a vendor's product contains software that is not part of the TOE and has not been evaluated. The Installation, Generation and Startup (IGS) documentation ensures that once the administrator has followed the installation and configuration guidance the result is a OTE in a secure configuration.
	AGD_ADM.1	The AGD_ADM.1 requirement mandates the developer provide the administrator with guidance on how to operate the TOE in a secure manner. This includes describing the interfaces the administrator uses in managing the TOE, security parameters that are configurable by the administrator, how to configure the TOE's rule set and the implications of any dependencies of individual rules.
	AGD_USR.1	The AGD_USR.1 requirement is intended for non-administrative users, but could be used to provide guidance on security that is common to both administrators and non-administrators (e.g., password management guidelines).
	AVA_MSU.2	AVA_MSU.2 ensures that the guidance documentation is complete and can be followed unambiguously to ensure the TOE is not mis-configured in an insecure state due to confusing guidance.

Objective	Requirements Addressing the Objective	Rationale
O.AUDIT_GENERATION	FAU_GEN.1-NIAP-0407 FAU_GEN.2-NIAP-0410 FIA_USB.1-NIAP-0415 FAU_SEL.1-NIAP-0407	FAU_GEN.1-NIAP-0407 defines the set of events that the TOE must be capable of recording. This requirement ensures that an administrator has the ability to audit any security relevant event that takes place in the TOE. This requirement also defines the information that must be contained in the audit record for each auditable event. There is a minimum of information that must be present in every audit record and this requirement defines that, as well as the additional information that must be recorded for each auditable event. This requirement also places a requirement on the level of detail that is recorded on any additional security functional requirements an ST author adds to this PP. FAU_GEN.2-NIAP-410 ensures that the audit records associate a user identity with the auditable event. Although the FIA_ATD.1 requirements mandate that a “userid” be used to represent a user identity, the TOE developer is able to associate different types of user-ids with different users in order to meet this objective. FAU_SEL.1-NIAP-0407 allows the selected administrator(s) to configure which auditable events will be recorded in the audit trail. This provides the administrator with the flexibility in recording only those events that are deemed necessary by site policy, thus reducing the amount of resources consumed by the audit mechanism and providing the ability to focus on

Objective	Requirements Addressing the Objective	Rationale
		the actions of an individual user. In addition, the requirement has been refined to require that the audit event selection function is configurable during run-time to ensure the TOE is able to capture security-relevant events given changes in threat conditions. FIA_USB.1 plays a role is satisfying this objective by requiring a binding of security attributes associated with users that are authenticated with the subjects that represent them in the TOE. This only applies to authenticated users, since the identity of unauthenticated users cannot be confirmed. Therefore, the audit trail may not always have the proper identity of the subject that causes an audit record to be generated (anonymous relying parties).
O.AUDIT_PROTECTION	FMT_MOF FAU_SAR.2 FAU_STG.1-NIAP-0429 FAU_STG.3 FAU_STG-NIAP-0414-1	FMT_MOF.1 restricts the ability to control the behavior of the audit and alarm mechanism to the Administrator. The Administrator is the only user that controls the behavior of the events that generate alarms and whether the alarm mechanism is enabled or disabled. FAU_SAR.2 restricts the ability to read the audit trail to the Auditor, thus preventing the disclosure of the audit data to any other user. However, the TOE is not expected to prevent the disclosure of audit data if it has been archived or saved in another form (e.g., moved or copied to an ordinary file).

Objective	Requirements Addressing the Objective	Rationale
		The FAU_STG family dictates how the audit trail is protected. FAU_STG.1-NIAP-0429 restricts the ability to delete audit records to the Auditor; or if the option of overwriting old audit records is chosen by the Platform/Directory Administrator in FAU_STG.NIAP-0414-1, the audit data may be deleted/overwritten. Since the auditor is trusted to review the audit data, the threat being countered is that the platform/directory administrator does something malicious and then attempts to conceal it by configuring the audit log to overwrite old records. Presumably the platform/directory administrator would then attempt to fill up the audit log in order to overwrite the thing they just did, as well as the fact that the they reconfigured the audit log overwrite action. The auditor would hopefully notice this activity and detect the fact that the platform/directory administrator was performing illicit activities. The fact that the platform/directory administrator does not directly have the ability to delete the audit records helps ensure that audit records are kept until the Auditor deems they are no longer necessary. FAU_STG.1-NIAP-0429 also ensures that no one has the ability to modify audit records (e.g., edit any of the information contained in an audit record). This ensures the integrity of the audit trail is maintained. FAU_STG.3 requires that the administrators be alerted when the

Objective	Requirements Addressing the Objective	Rationale
		audit trail exceeds a capacity threshold established by the Administrator. In addition, an audit record is cut which will trigger the analysis performed in FAU_SAA, resulting in an FAU_ARP alarm being issued. This ensures that an administrator has the opportunity to manage the audit trail before it becomes full and the avoiding the possible loss of audit data. FAU_STG.NIAP-0414-1 allows the Administrator to configure the TOE so that if the audit trail does become full, either the TOE will prevent any events from occurring (other than actions taken by the administrator) that would generate an audit record or the audit mechanism will overwrite the oldest audit records with new records. FMT_SMF.1 requires the TOE to provide an administrator with a facility to backup, recover and archive audit data ensuring the ability to recover corrupted audit records, and access to a complete history of audit information.
O.AUDIT_REVIEW The TOE will provide the capability to selectively view audit information, and alert the administrator of identified potential security violations.	FAU_ARP.1 FAU_ARP_ACK_(EXP).1 FAU_SAA.1-NIAP-0407 FAU_SAR.1 FAU_SAR.3	FAU_SAA.1-NIAP-0407 defines the events (or rules) that indicate a potential security violation and will generate an alarm. The triggers for these events are largely configurable by the Administrator. Some rules are not configurable, or configurable by the administrator. FAU_ARP.1 requires that the alarm be displayed at the local administrative console. FAU_ARP_ACK_(EXP).1 requires

Objective	Requirements Addressing the Objective	Rationale
		that an alarm generated by the mechanism that implements the FAU_ARP requirement be maintained until an administrator acknowledges it. This ensures that the alarm message will not be obstructed and the administrators will be alerted of a potential security violation. Additionally, this requires that the acknowledgement be transmitted to users that received the alarm, thus ensuring that that set of administrators knows that the user specified in the acknowledgement message has addressed the alarm. FAU_SAR.1 (both iterations) is used to provide both the auditor and an external audit analysis function the capability to read all the audit data contained in the audit trail. This requirement also mandates the audit information be presented in a manner that is suitable for the end user (auditor or external system) to interpret the audit trail. It is expected that the audit information be presented in such a way that the end user can examine an audit record and have the appropriate information (that required by FAU_GEN.2-NIAP-410) presented together to facilitate the analysis of the audit review. Ensuring the audit data are presented in an interpretable format will enhance the ability of the entity performing the analysis to identify potential security violations. FAU_SAR.3 complements FAU_SAR.1 by providing the administrators the flexibility to specify criteria that can be used to

Objective	Requirements Addressing the Objective	Rationale
		search or sort the audit records residing in the audit trail. FAU_SAR.3 requires the administrators be able to establish the audit review criteria based on a userid and role so that the actions of a user can be readily identified and analyzed. Allowing the administrators to perform searches or sort the audit records based on dates and times provides the capability to facilitate the administrator's review of incidents that may have taken place at a certain time. It is important to note that the intent of sorting in this requirement is to allow the administrators the capability to organize or group the records associated with a given criteria.
O.CHANGE_MANAGEMENT The configuration of, and all changes to, the TOE and its development evidence will be analyzed, tracked, and controlled throughout the TOE's development.	ACM_AUT.1	ACM_AUT.1 complements ACM_CAP.4 by requiring that the CM system use an automated means to control changes made to the TOE. If automated tools are used by the developer to analyze, or track changes made to the TOE, those automated tools must be described. This aids in understanding how the CM system enforces the control over changes made to the TOE.

Objective	Requirements Addressing the Objective	Rationale
	ACM_CAP.4	ACM_CAP.4 contributes to this objective by requiring the developer have a configuration management plan that describes how changes to the TOE and its evaluation deliverables are managed. The developer is also required to employ a configuration management system that operates in accordance with the CM plan and provides the capability to control who on the development staff can make changes to the TOE and its development evidence. This requirement also ensures that authorized changes to the TOE have been analyzed and the developer's acceptance plan describes how this analysis is performed and how decisions to incorporate the changes to the TOE are made.
	ACM_SCP.2	ACM_SCP.2 is necessary to define what items must be under the control of the CM system. This requirement ensures that the TOE implementation representation, design documentation, test documentation (including the executable test suite), user and administrator guidance, CM documentation and security flaws are tracked by the CM system.

Objective	Requirements Addressing the Objective	Rationale
	ALC_DVS.1	ALC_DVS.1 requires the developer to describe the security measures they employ to ensure the integrity and confidentiality of the TOE are maintained. They physical, procedural, and personnel security measures the developer uses provides an added level of control over who and how changes are made to the TOE and its associated evidence.
	ALC_FLR.2	ALC_FLR.2 plays a role in satisfying the “analyzed” portion of this objective by requiring the developer to have procedures that address flaws that have been discovered in the product, either through developer actions (e.g., developer testing) or those discovered by others. The flaw remediation process used by the developer corrects any discovered flaws and performs an analysis to ensure new flaws are not created while fixing the discovered flaws.
	ALC_LCD.1	ALC_LCD.1 requires the developer to document the life-cycle model used in the development and maintenance of the TOE. This life-cycle model describes the procedural aspects regarding the development of the TOE, such as design methods, code or documentation reviews, how changes to the TOE are reviewed and accepted or rejected.

Objective	Requirements Addressing the Objective	Rationale
O.CORRECT_TSF_OPERATION The TOE will provide a capability to test the TSF to ensure the correct operation of the TSF in its operational environment.	FPT_TST_(EXP).4	O.CORRECT_TSF_OPERATION requires two security functional requirements in the FPT class, FPT_TST. These functional requirements provide the end user with the capability to ensure the TOE's security mechanisms continue to operate correctly in the field. FPT_TST_(EXP).4 has been created to ensure end user tests exist to demonstrate the correct operation of the security mechanisms required by the TOE that are provided by the hardware and that the TOE's software and TSF data has not been corrupted. Hardware failures could render a TOE's software ineffective in enforcing its security policies and this requirement provides the end user the ability to discover any failures in the hardware security mechanisms.
	FPT_TST_(EXP).5	FPT_TST_(EXP).5 is necessary to ensure the cryptographic components of the TSF are functioning correctly. This will be done by using the suite of self-tests provided by the FIPS 14-2 cryptographic module. The administrative role may invoke self-tests at any time. These self-tests will also be performed at selected intervals.

Objective	Requirements Addressing the Objective	Rationale

O.CRYPTOGRAPHIC_ FUNCTIONS
The TOE shall provide cryptographic functions for its own use, including encryption/decryption and digital signature operations.
	FCS_CKM.1(1)
	FCS_CKM.1(2)
	FCS_CKM.2
	FCS_CKM.4
	FCS_CKM_(EXP).1
	FCS_CKM_(EXP).2
	FCS_COA_(EXP).1
	FCS_COP.1(1)
	FCS_COP.1(2)
	FCS_COP.1(3)
	FCS_COP.1(4)
	FCS_COP_(EXP).1
		The FCS requirements used in this PP satisfy this objective by levying requirements that ensure the cryptographic standards include the NIST FIPS publications (where possible) and NIST approved ANSI standards. The intent is to have the satisfaction of the cryptographic standards be validated through a NIST FIPS 140 validation.
		In contrast to O.CRYPTOGRAPHY_VALIDATED, this objective is to provide cryptographic functionality that is used by the TOE. The core functionality to be supported is encryption/decryption using a symmetric algorithm. Since these operations involve cryptographic keys, how the keys are generated and/or otherwise obtained have to also be specified.

	FCS_CKM.1	FCS_CKM.1(1) is a requirement that a cryptomodule generate symmetric keys. FCS_CKM.1(2) requirement specifies that asymmetric cryptographic keys (a public and private key pair) be generated for the PDA. This partly satisfies O.PDA_PKI since this will create a private key for the PDA, and a public key which can be distributed, thus, enabling the PDA to take advantage of PKI.
	FCS_CKM.2	FCS_CKM.2 requires the TSF to distribute cryptographic keys in accordance with a prescribed method.

Objective	Requirements Addressing the Objective	Rationale
	FCS_CKM.4	FCS_CKM.4 provides the functionality for ensuring key and key material is zeroized. This applies not only to key that resides in the TOE, but also to intermediate areas (physical memory, page files, memory dumps, etc.) where keys may appear.
	FCS_CKM_(EXP).1 FCS_CKM_(EXP).2	FCS_CKM_(EXP).1 states the TSF must validate all symmetric and asymmetric keys to ensure no weak keys are used. FCS_CKM_(EXP).2 states how the TSF must handle and store all cryptographic keys.
	FCS_COA_(EXP).1	FCS_COA_(EXP).1 states that encryption and decryption operations must be available for applications on the TOE. This partly satisfies O.PDA_PKI by allowing applications to leverage the use of certificates.

Objective	Requirements Addressing the Objective	Rationale

	FCS_COP.1(1)
	FCS_COP.1(2)
	FCS_COP.1(3)
	FCS_COP.1(4)
		FCS_COP.1(1) states the operations that must be followed for encryption and decryption. It must use TDEA encryption with key size of 168 bits.
		FCS_COP.1(2) specifies encryption algrothms that must be used to secure cryptographic signature services.
		FCS_COP.1(3) requires the TOE to perform cryptographic key transport services.
		FCS_COP.1(4) specifies encryption algorithms that must be used to secure cryptographic signature services and cryptographic key agreements services.

	FCS_COP_(EXP).1	FCS_COP_(EXP).1 requires the TSF to perform random number generation services.

Objective	Requirements Addressing the Objective	Rationale
O.CRYPTOGRAPHY_ VALIDATED The TOE shall use NIST FIPS 140-2 validated cryptomodules for cryptographic services implementing FIPS-approved security functions and random number generation services used by cryptographic functions.	FCS_BCM_(EXP).1 FCS_COP_(EXP).1	This objective deals with the issue of using FIPS 140-2-approved cryptomodules in the TOE. A cryptomodule, as used in the components, is a module that is FIPS 140-2 validated (in accordance with FCS_BCM_(EXP).1); the cryptographic functionality implemented in that module are FIPS-approved security functions that have been validated; and the cryptographic functionality is available in a FIPS-approved mode of the cryptomodule. This objective is distinguished from O.CRYPTOGRAPHIC_FUNCTIONS in that this deals only with a requirement to use FIPS 140-2-validated cryptomodules where the TOE requires such functionality; it does not dictate the specific functionality that is to be used. FCS_BCM_(EXP).1 is an explicit requirement that specifies not only that cryptographic functions that are FIPS-approved must be validated by FIPS, but also what NIST FIPS rating level the cryptographic module must satisfy. The level specifies the degree of testing of the module. The higher the level, the more extensive the module is tested. FCS_COP_(EXP).1 specifies that the random number generator must be FIPS-approved and validated. It must also use NIST-approved hashing functions.

Objective	Requirements Addressing the Objective	Rationale
O.DISPLAY_BANNER The TOE will display an advisory warning regarding use of the TOE	FTA_TAB.1	FTA_TAB.1 meets this objective by requiring the TOE display an administrator-defined banner before the user initiates a session. This banner is under complete control of the Administrator in which they specify any warnings regarding unauthorized use of the TOE and remove any product or version information if they desire.
O.DOCUMENT_KEY_ LEAKAGE	AVA_CCA_(EXP).2	AVA_CCA_(EXP).2 requires that a covert channel analysis be performed on the entire TOE to determine the bandwidth of possible cryptographic key leakage. While there are no requirements to limit the bandwidth, the results of this analysis will provide useful guidance on what the specified lifetime of the cryptographic keys should be in order to reduce the damage due to a key compromise.
O.ENCRYPT_STORED _DATA All user data stored within the TOE will be encrypted using TDEA with a key size of 168 bits.	FCS_COP.1(1)	FCS_COP.1(1) specifies the algorithms that must be used to encrypt and decrypt all user data. This is to ensure that the algorithms used are strong enough to thwart an attack of medium threat level.
	FDP_ACF.1	FDP_ACF.1 defines the security attributes and the rules for the Stored Data Policy. The rules state that all user data stored within the PED must be encrypted using NIST-approved algorithms and key sizes.

Objective	Requirements Addressing the Objective	Rationale
	FDP_ACC.2	FDP_ACC.2 defines the subjects and objects that will be used in the Stored Data Policy. The subjects consist of all the process which store to and retrieve from persistent memory in the PED. Objects are all forms of user data that is stored on the PED.
O.MAINT_MODE The TOE shall provide a mode from which recovery or initial startup procedures can be performed	FPT_RCV.2	This objective is met by using the FPT_RCV.2 requirement, which ensures that the TOE does not continue to operate in an insecure state when a hardware or software failure occurs. Upon the failure of the TSF self-tests the TOE will no longer be assured of enforcing its security policies. Therefore, the TOE enters a state that operations cease and requires an administrator to follow documented procedures that instruct them on to return the TOE to a secure state. These procedures may include running diagnostics of the hardware, or utilities that may correct any integrity problems found with the TSF data or code. Solely specifying that the administrator reload and install the TOE software from scratch, while might be required in some cases, does not meet the intent of this requirement.

Objective	Requirements Addressing the Objective	Rationale
O.MANAGE The TOE will provide all the functions and facilities necessary to support the administrators in their management of the security of the TOE, and restrict these functions and facilities from unauthorized use.	FMT_MTD.1	The FMT requirements are used to satisfy this management objective, as well as other objectives that specify the control of functionality. The requirement's rationale for this objective focuses on the administrator's capability to perform management functions in order to control the behavior of security functions.
	FMT_MOF.1(1)	There are several functions in the TSF that need to be enabled or disabled. The use of the security functions is specified and restricted by the FMT_MOF.1 iterations. FMT_MOF.1(1) allows the Administrator to set the time interval for which all non-cryptographic self-tests will run.
	FMT_MOF.1(2)	FMT_MOF.1(2) allows the Administrator to set the time interval for which all cryptographic self-tests will run.
	FMT_MOF.1(3)	FMT_MOF.1(3) allows all administrative roles to invoke any self-test at anytime they wish. This allows the administrator to test the TSF if they notice something working incorrectly instead of waiting for the next pre determined time for the self-tests to run.

Objective	Requirements Addressing the Objective	Rationale
	FMT_SMF.1	The requirement FMT_SMF.1 was introduced as an international interpretation. This requirement specifies functionality that must be provided to administrators of the TOE.
O.MEDIATE The TOE must protect user data in accordance with its security policy.	FDP_ACC.2	FDP_ACC.2 defines the subjects, objects and operations of the access control policy.
	FDP_ACF.1	FDP_ACF.1 defines the rules for subjects and objects in the Stored Data Policy. This policy ensures that data stored on the TOE in persistent memory is encrypted and decrypted using the specified algorithm and key size.
	FDP_IFC.1	FDP_IFC.1 defines the subjects, information and operations of the External Flow Policy.

	FDP_IFC.2(1)
	FDP_IFC.2(2)
		FDP_IFC.2(1) and FDP_IFC.2(2) define the subjects, information (e.g., objects) and the operations that are performed with respect to the two information flow policies.

	FDP_IFF.1-NIAP-0407(1)	FDP_IFF.1-NIAP-0407(1) defines the rules for subjects and objects in the Application Separation Policy. Each application is only allowed to view or modify data in its domain. This will ensure a malicious program cannot modify data another application created, such as the users personal address book.

Objective	Requirements Addressing the Objective	Rationale
	FDP_IFF-NIAP-0407(2)	FDP_IFF.1-NIAP-0407(2) defines the rules for the SCIF Mode Policy. This policy states that no information may leak into or out of the PED device while it is located in a SCIF. The user must manually put the device into and out of this mode. While in this mode the TOE must not leak data from the TOE or collect data to be stored on the TOE. For example, the TOE cannot record a conversation inside of a SCIF.
	FDP_IFF-NIAP-0407(3)	FDP_IFF.1-NIAP-0407(3) defines the rules for subjects and objects of the External Flow Policy. This policy ensures that the TOE will communicate securely through the use of encryption with secure remote server for which it has the verified digital signature.
O.REPLAY_DETECTION The TOE will provide a means to detect and reject the replay of authentication data, as well as TSF data and security attributes.	FPT_RPL.1	The O.REPLAY objective is satisfied by FPT_RPL.1, which requires the TOE to detect and reject the replay of authentication data, as well as TSF data and security attributes.
O.RESIDUAL_INFORMATION	FDP_RIP.2	FDP_RIP.2 counters this threat by ensuring that TSF data and user data is not persistent when resources are released by one user/process and allocated to another user/process.
O.RESOURCE_SHARING The TOE shall provide mechanisms that mitigate attempts to exhaust resources provided by	FRU_RSA.1 FMT_MTD.2 FMT_MOF.1	The following are examples of iterations of FMT_MTD.1 and FRU_RSA.1 that were used by Protection Profile authors to satisfy some of the functions of

Objective	Requirements Addressing the Objective	Rationale
resources provided by the TOE.		O.RESOURCE_SHARING: While an availability security policy does not explicitly exist, FRU_RSA.1 is used to mitigate potential resource exhaustion attempts. In order to mitigate the CPU exhaustion attempt, FRU_RSA.1(1) is included. This requires that the CPU time being consumed by a relying party must be limited to an amount specified by the administrator (FMT_MTD.2), and actions taken when an attempt is made are specified in FMT_MTD.2. This requirement takes into account all CPU resources being consumed by a user (relying party), and not just a single subject. FRU_RSA.1(2) was used to reduce the impact of an attempt being made to exhaust transport-layer representation implementation artifacts (e.g., the TCP "half-open connection" attack). This requirement indicates that a time period must exist when maximum quota (which is defined by the ST) is met or surpassed. Although this requirement (unlike the two previous requirements) does not mandate that the administrator be able to set this time period, FMT_MTD.2 restricts this functionality should the TOE implement it. FMT_MTD.2 also indicates (when filled in by the ST author) what action is to be taken when the quota is reached. FMT_MOF.1 dictates the functionality required to manage the security functions of the TOE. The

Objective	Requirements Addressing the Objective	Rationale
		ability to control this function is limited to the Administrator and provides this role the capability of enabling or disabling the function. This requirement also provides the Administrator with the capability to modify the behavior of the function that indicates a potential sharing violation. So as to ensure the mechanisms are configured as intended, the Administrator has the ability to view the conditions under which an sharing alarm will be generated, and if alarm generation is enabled.
O.ROBUST_TOE_ACCESS The TOE will provide mechanisms that control a user's logical access to the TOE and to explicitly deny access to specific users when appropriate	FIA_AFL.1	FIA_AFL.1 provides a detection mechanism for unsuccessful authentication attempts. The requirement enables an Administrator settable threshold that prevents unauthorized users from gaining access to an authorized user's account by guessing authentication data by locking the targeted account for some Administrator defined time period. Thus, limiting an unauthorized user's ability to gain unauthorized access to the TOE.
	FIA_ATD.1	FIA_ATD.1 defines the attributes of users, including a userid that is used by the TOE to determine a user's identity and enforce what type of access the user has to the TOE (e.g., the TOE associates a userid with any role(s) they may assume).

Objective	Requirements Addressing the Objective	Rationale
	FIA_UAU.2	FIA_UAU.2 requires that the user authenticate themselves to the TOE before performing administrative duties or using the services identified in this requirement.
	FIA_UID.2	FIA_UID.2 plays a small role in satisfying this objective by ensuring that every user is identified before the TOE performs any mediated functions.
	FIA_USB.1-NIAP-0415	FIA_USB.1-NIAP-0415 ensures the TOE will bind all security attributes with the authorized user, this includes but is not limited to the password and administrative roles.
	FTA_SSL.1	The FTA_SSL family partially satisfies the O.ROBUST_TOE_ACCESS objective by ensuring that a user's sessions are afforded some level of protection. FTA_SSL.1 provides the Administrator the capability to specify a time interval of inactivity in which an unattended session would be locked and will require the user to re-authenticate before the session can be used to access TOE resources.
	FTA_SSL.2	FTA_SSL.2 provides users the ability to lock their session. This component allows users to protect their session immediately, rather then waiting for the time-out period and minimizes their session's risk of exposure.

Objective	Requirements Addressing the Objective	Rationale
	AVA_SOF.1	The AVA_SOF.1 requirement is applied to the local authentication mechanism. For this TOE, the strength of function specified is medium. This requirement ensures the developer has performed an analysis of the authentication mechanism to ensure the probability of guessing the user's authentication data would require high-attack potential, as defined in Annex B of the CEM.
O.SELF_PROTECTION The TSF will maintain a domain for its own execution that protects itself and its resources from external interference, tampering or unauthorized disclosure.	FPT_SEP.2	FPT_SEP.2 was chosen to ensure the TSF provides a domain that protects itself from untrusted users. If the TSF cannot protect itself it cannot be relied upon to enforce its security policies. FPT_SEP.1 could have been used to address the previous notio n, however, FPT_SEP.2 was used to require that the cryptographic module be provided its own address space. This is necessary to reduce the impact of programming errors in the remaining portions of the TSF on the cryptographic module.

Objective	Requirements Addressing the Objective	Rationale
	FPT_RVM.1	The inclusion of FPT_RVM.1 ensures that the TSF makes policy decisions on all interfaces that perform operations on subjects and objects that are scoped by the policies. Without this non-bypassability requirement, the TSF could not be relied upon to completely enforce the security policies, since an interface(s) may otherwise exist that would provide a user with access to TOE resources (including TSF data and executable code) regardless of the defined policies. This includes controlling the accessibility to interfaces, as well as what access control is provided within the interfaces.

Objective	Requirements Addressing the Objective	Rationale
O.SOUND_DESIGN The TOE will be designed using sound design principles and techniques. The TOE design, design principles and design techniques will be adequately and accurately documented.	ADV_INT_(EXP).1	There are two different perspectives for this objective. One is from the developer's point of view and the other is from the evaluator's. The ADV class of requirements is levied to aide in the understanding of the design for both parties, which ultimately helps to ensure the design is sound. ADV_INT_(EXP).1 ensures that the design of the TOE has been performed using good software engineering design principles that require a modular design of the TSF. Modular code increases the developer's understanding of the interactions within the TSF, which in turn, potentially reduces the amount of errors in the design. Having a modular design is imperative for evaluator's to gain an appropriate level of understanding of the TOE's design in a relatively short amount of time. The appropriate level of understanding is dictated by other assurance requirements in this PP (e.g., ATE_SPT.2, AVA_CCA_(EXP).2, AVA_VLA.3)
	ADV_SPM.1	ADV_SPM.1 requires the developer to provide an informal model of the security policies of the TOE. Modeling these policies helps understand and reduce the unintended side effects that occur during the TOE's operation that might adversely affect the TOE's ability to enforce its security policies.

Objective	Requirements Addressing the Objective	Rationale
	ADV_FSP_(EXP).1	ADV_FSP_(EXP).1 requires that the interfaces to the TSF be completely specified. In this TOE, a complete specification of the external interface is critical in understanding what functionality is presented to untrusted users and how that functionality fits into the enforcement of security policies. The functional specification of the hardware interface is also extremely critical. Any processing that is externally visible performed by an external interface must be specified in the functional specification. Having a complete understanding of what is available at the TSF interface allows one to analyze this functionality in the context of design flaws.
	ADV_HLD_(EXP).1	ADV_HLD_(EXP).1 requires that a high-level design of the TOE be provided. This level of design describes the architecture of the TOE in terms of subsystems. It identifies which subsystems are responsible for making and enforcing security relevant (e.g., anything relating to an SFR) decisions and provides a description, at a high level, of how those decisions are made and enforced. Having this level of description helps provide a general understanding of how the TOE works, without getting buried in details, and may allow the reader to discover flaws in the design.

Objective	Requirements Addressing the Objective	Rationale
	ADV_ARC_(EXP).1	ADV_ARC_(EXP).1 addresses the non-bypassability (FPT_RVM.1) and domain separation (FPT_SEP.2) aspects of the TSF, since these need to be analyzed differently from other functional requirements.
	ADV_LLD_(EXP).1	The low-level design, as required by ADV_LLD_(EXP).1, provides the reader with the details of the TOE's design and describes at a module level how the design of the TOE addresses the SFRs. This level of description provides the detail of how modules interact within the TOE and if a flaw exists in the TOE's design, it is more likely to be found here rather than the high-level design. This requirement also mandates that the interfaces presented by modules be specified. Having knowledge of the parameters a module accepts, the errors that can be returned and a description of how the module works to support the security policies allows the design to be understood at its lowest level.
	ADV_RCR.1	ADV_RCR.1 is used to ensure that the levels of decomposition of the TOE's design are consistent with one another. This is important, since design decisions that are analyzed and made at one level (e.g., functional specification) that are not correctly designed at a lower level may lead to a design flaw. This requirement helps in the design analysis to ensure design decisions are realized at all levels of the design.

Objective	Requirements Addressing the Objective	Rationale
O.SOUND_IMPLEMEN TATION The implementation of the TOE will be an accurate instantiation of its design, and is adequately and accurately documented	ADV_LLD_(EXP).1 ADV_ARC_(EXP).1	While ADV_LLD_(EXP).1 (and ADV_ARC_(EXP).1 for the FPT_SEP.2 and FPT_RVM.1 aspects of the TSF) is used to aide in ensuring that the TOE's design is sound, it also contributes to ensuring the implementation is correctly realized from the design. It is expected that evaluators will use the low-level design a san aide in understanding the implementation representation. The low-level design requirements ensure the evaluators have enough information to intelligently analyze (e.g., the documented interface descriptions of the modules match the entry points in the module, error codes returned by the functions in the module are consistent with those identified in the documentation) the implementation and ensure it is consistent with the deign.

Objective	Requirements Addressing the Objective	Rationale
	ADV_IMP.2	While evaluators have the ability to "negotiate" the subset in ADV_IMP.1, ADV_IMP.2 was chosen to ensure evaluators have full access to the source code. If the evaluators are limited in their ability to analyze source code they may not be able to determine the accuracy of the implementation or the adequacy of the documentation. Often times it is difficult for an evaluator to identify the complete sample of code they with to analyze. Often times looking at code in one subsystem may lead the evaluator to discover code they should look at in another subsystem. Rather than require the evaluator to "re-negotiate" another sample of code, the complete implementation representation is required.
	ADV_INT_(EXP).1	When performing the activities associated with the ADV_INT_(EXP).1 requirement, the evaluators will ensure that the architecture of the implementation is modular and consistent with the architecture presented in the low-level design. Having a modular implementation provides the evaluators with the ability to more easily assess the accuracy of the implementation, with respect to the design. If the implementation is overly complex (e.g., circular dependencies, not well understood coupling, reliance on side-effects) the evaluator may not have the ability to assess the accuracy of the implementation.

Objective	Requirements Addressing the Objective	Rationale
	ALC_TAT.1	ALC_TAT.1 provides evaluators with information necessary to understand the implementation representation and what the resulting implementation will consist of. Critical areas (e.g., the use of libraries, what definitions are used, compiler options) are documented so the evaluator can determine how the implementation representation is to be analyzed.
	ADV_RCR.1	ADV_RCR.1 is used here to provide the correspondence of the lowest level of decomposition (e.g., source code) to the adjoining level, low-level design. The correspondence analysis is used by the evaluator as a tool when determining if the low-level design is correctly reflected in the implementation representation.
O.THOROUGH_FUNCTIONAL_TESTING The TOE will undergo appropriate security functional testing that demonstrates the TSF satisfies the security functional requirements.	ATE_FUN.1	In order to satisfy O.THOROUGH_FUNCTIONAL_TESTING, the ATE class of requirements is necessary. The component ATE_FUN.1 requires the developer to provide the necessary test documentation to provide the necessary test documentation to allow for an independent analysis of the developer's security functional test coverage. In addition, the developer must provide the test suite executables and source code, which are used for independently verifying the test suite results and in support of the test coverage analysis activities.

Objective	Requirements Addressing the Objective	Rationale
	ATE_COV.2	ATE_COV.2 requires the developer to provide a test coverage analysis tat demonstrates the TSFI and are completely addressed by the developer's test suite. While exhaustive testing of the TSFI is not required, this component ensures that the security functionality of each TSFI is addressed. This component also requires an independent confirmation of the completeness of the test suite, which aids in ensuring that correct security relevant functionality of a TSFI is demonstrated through the testing effort.
	ATE_DPT.2	ATE_DPT.2 requires the developer to provide a test coverage analysis that demonstrates depth of coverage of the test suite. This component complements ATE_COV.2 by ensuring that the developer takes into account the high-level and low-level design when developing their test suite. Since exhaustive testing of the TSFI is not required, ATE_DPT.2 ensures that subtleties in TSF behavior that are not readily apparent in the function specification are addressed in the test suite.

Objective	Requirements Addressing the Objective	Rationale
	ATE_IND.2	ATE_IND.2 requires an independent confirmation of the developer’s test results, by mandating a subset of the test suite be run by an independent party. This component also requires an independent party to attempt to craft functional tests that address functional behavior that is not demonstrated in the developer’s test suite. Upon successful adherence to these requirements, the TOE’s conformance to the specified security functional requirements will have been demonstrated.
O.TIME_STAMPS	FPT_STM.1 FMT_MTD.1	FPT_STM.1 requires that the TOE be able to provide reliable time stamps for its own use and therefore, partially satisfies this objective. Time stamps include date and time and are reliable in that they are always available to the toe, and the clock must be monotonically increasing. FMT_MTD.1 satisfies the rest of this objective by providing the capability to set the time used for generating time stamps to the administrator.

Objective	Requirements Addressing the Objective	Rationale
O.TRUSTED_PATH The TOE will provide a means to ensure that users are not communicating with some other entity pretending to be the TOE when supplying identification and authentication data.	FTP_TRP.1	FTP_TRP.1.1 requires the TOE to provide a mechanism that creates a distinct communication path that protects the data that traverses this path from disclosure or modification. This requirement ensures that the TOE can identify the end points and ensures that a user cannot insert themselves between the user and the TOE, by requiring that the means used for invoking the communication path cannot be intercepted and allow a "man-in-the-middle-attack" (this does not prevent someone from capturing the traffic and replaying it at a later time – see FPT_RPL.1). Since the user invokes the trusted path (FTP_TRP.1.2) mechanism they can be assured they are communicating with the TOE. FTP_TRP.1.3 mandates that the trusted path be the only means available for providing identification and authentication information, therefore ensuring a user's authentication data will not be compromised when performing authentication functions.
O.USER_GUIDANCE The TOE will provide users with the information necessary to correctly use the security mechanisms	AGD_USR.1	The user guidance required by AGD_USR.1 meets the objective by describing the access controls available to the user, and how to set the attributes pertaining to the mechanism. This guidance also instructs the user how to log on to the TOE, and how to choose passwords that will not be easily compromised through a brute force attack.

Objective	Requirements Addressing the Objective	Rationale
O.VULNERABILITY_ ANALYSIS_TEST The TOE will undergo appropriate independent vulnerability analysis and penetration testing to demonstrate the design and implementation of the TOE does not allow attackers with medium attack potential to violate the TOE's security policies	AVA_VLA.3	To maintain consistency with the overall assurance goals of this TOE, O.VULNERABILITY_ANALYSIS _TEST requires the AVA_VLA.3 component to provide the necessary level of confidence that vulnerabilities do not exist in the TOE that could cause the security policies to be violated. AVA_VLA.3 requires the developer to perform a systematic search for potential vulnerabilities in all the TOE deliverables. For those vulnerabilities that are not eliminated, a rationale must be provided that describes why these vulnerabilities cannot be exploited by a threat agent with a moderate attack potential, which is in keeping with the desired assurance level of this TOE. As with the function testing, a key element in this component is that an independent assessment of the completeness of the developer's analysis is made, and more importantly, an independent vulnerability analysis coupled with testing of the TOE is performed. This component proved the confidence that security flaws do not exist in the TOE that could be exploited by a threat agent of moderate (or lower) attack potential to violate the TOE's security policies.

6.3.1 Cellular Communications Package Security Requirements Rationale

The rationale listed here for each of the Objectives must be combined with the Rationale Table in the base PED PP to achieve a completed mapping. Only those Objectives with the stated Requirements are listed in this table.

Table 17 Security Objective to Functional Component Mapping Rationale for Cellular Communications

Objective	Requirements Addressing the Objective	Rationale
O.MEDIATE	FDP_IFC[CELL].1(A)	FDP_IFC[CELL].1(A) defines the subjects, objects and processes of the Voice Transport Protection Policy.
	FDP_IFC[CELL].1(B)	FDP_IFC[CELL].1(B) defines the subjects, objects and processes of the SMS Transport Protection Policy.
	FDP_IFF[CELL].1-NIAP-0407(A)	FDP_IF[CELL]F.1-NIAP-0407(A) defines the rules of the Voice Transport Protection Policywhich ensures that the TOE is able to utilize valid digital certificates to send encrypted, signed email messages.
	FDP_IFF[CELL].1-NIAP-0407(B)	FDP_IFF[CELL].1-NIAP-0407(B) defines the rules of the SMS Transport Protection Policy which ensures that the TOE is able to utilize valid digital certificates to send encrypted, signed email messages.

6.4 Rationale for Assurance Requirements

135 The EAL definitions and assurance requirements in Part 3 of the CC were reviewed and the *Medium Robustness Assurance Package* as defined in Section 5.3 was believed to best achieve the goal of addressing circumstances where developers and users require a moderate to high level of independently assured security in commercial products. The assurance package selection was based on:

a) recommendations documented in the Global Information Grid (GIG);
b) Department of Defense (DoD) Instruction 8500.1; and
c) the postulated threat environment.

136 This collection of assurance requirements require TOE developers to gain assurance from good software engineering development practices which, though rigorous, do not require substantial specialist knowledge, skills, and other resources. Rationale for individual assurance requirements is provided in Table 16.

137 The Government's guidance in the GIG was consulted and found to also support the chosen assurance package. Specifically, the GIG states that medium robustness security services and mechanisms provide for additional safeguards above the Department of Defense (DoD) minimum and require good assurance security design as specified in Evaluation Assurance Level (EAL)3 or greater.

138 The postulated threat environment specified in Section 3 of this PP was used in conjunction with the Information Assurance Technical Framework (IATF) Robustness Strategy guidance to derive the chosen assurance level.

139 These three factors were taken into consideration and the conclusion was that the medium robustness assurance package was the appropriate level of assurance.

6.5 Rationale for Strength of Function Claim

140 Part 1 of the CC defines "strength of function" in terms of the minimum efforts assumed necessary to defeat the expected security behavior of a TOE security function. There are three strength of function levels defined in Part 1: SOF-basic, SOF-medium and SOF-high. SOF-medium is the strength of function level chosen for this PP. SOF-medium states, "a level of the TOE strength of function where analysis shows that the function provides adequate protection against straightforward or intentional breach of TOE security by attackers possessing a moderate attack potential." The rationale for choosing SOF-medium was to be consistent with the TOE objective O.VULNERABILITY_ANALYSIS_TEST and assurance requirements included in this PP. Specifically, AVA_VLA.3 requires that the TOE be resistant to an attacker with a moderate-attack potential, this is consistent with SOF-medium. Consequently, the metrics (i.e., passwords and keys) chosen for inclusion in this PP were determined to be acceptable for SOF-medium and would adequately protect information in a Medium Robustness Environment.

6.6 Rationale for Satisfying all Dependencies

Table 18 Functional Requirement Dependencies

Requirement	Dependency	Satisfied
FAU_ARP.1	FAU_SAA.1	Yes
FAU_ARP_ACK_(EXP).1	None	N/A
FAU_GEN.1-NIAP-0407	FPT_STM.1	Yes
FAU_GEN.2-NIAP-0410	FAU_GEN.1 FIA_UID.1	Yes, FIA_UID.2 satisfies the FIA_UID.1 dependency.
FAU_SAA.1-NIAP-0407	FAU_GEN.1	Yes
FAU_SAR.1	FAU_GEN.1	Yes
FAU_SAR.2	FAU_SAR.1	Yes
FAU_SAR.3	FAU_SAR.1	Yes
FAU_SEL.1-NIAP-0407	FAU_GEN.1 FMT_MTD.1	Yes
FAU_STG.1-NIAP-0429	FAU_GEN.1	Yes
FAU_STG.3	FAU_STG.1	Yes
FAU_STG.NIAP-0414	FMT_MOF or FMT_MTD	Yes
FCS_BCM_(EXP).1	None	N/A
FCS_CKM.1(1)	[FCS_CKM.2 or FCS_COP.1] FCS_CKM.4 FMT_MSA.2[24]	FCS_COP.1(1) satisfies the dependency on FCS_COP.1. FCS_CKM.4 satisfies the other dependency.

[24] The FMT_MSA.2 dependency is satisfied by FCS_CKM_(EXP).1. This requirement validates each generated key in accordance with FIPS standards to ensure weak keys are not used.

Requirement	Dependency	Satisfied
FCS_CKM.1(2)	[FCS_CKM.2 or FCS_COP.1] FCS_CKM.4 FMT_MSA.2[24]	FCS_COP.1(1) satisfies the dependency on FCS_COP.1. FCS_CKM.4 satisfies the other dependency.
FCS_CKM.2	[FDP_ITC.1 or FCS_CKM.1] FCS_CKM.4 FMT_MSA.2[24]	Yes, FCS_CKM.1 and FCS_CKM.4 satisfy the dependencies.
FCS_CKM.4	[FDP_ITC.1 or FCS_CKM.1] FMT_MSA.2[24]	Yes, FCS_CKM.1 satisfies the dependency.
FCS_CKM_(EXP).1	None	N/A
FCS_CKM_(EXP).2	None	N/A
FCS_COA_(EXP).1	None	N/A
FCS_COP.1(1)	[FDP_ITC.1 or FCS_CKM.1] FCS_CKM.4 FMT_MSA.2[24]	Yes, FCS_CKM.1 and FCS_CKM.4 satisfy the dependencies.
FCS_COP.1(2)	[FDP_ITC.1 or FCS_CKM.1] FCS_CKM.4 FMT_MSA.2[24]	Yes, FCS_CKM.1 and FCS_CKM.4 satisfy the dependencies.
FCS_COP.1(3)	[FDP_ITC.1 or FCS_CKM.1] FCS_CKM.4 FMT_MSA.2[24]	Yes, FCS_CKM.1 and FCS_CKM.4 satisfy the dependencies.

Requirement	Dependency	Satisfied
FCS_COP.1(4)	[FDP_ITC.1 or FCS_CKM.1] FCS_CKM.4 FMT_MSA.2[24]	Yes, FCS_CKM.1 and FCS_CKM.4 satisfy the dependencies.
FCS_COP_(EXP).1	None	N/A
FDP_ACC.2(1)	FDP ACF.1	Yes
FDP_ACC.2(2)	FDP ACF.1	Yes
FDP_ACF.1(1)	FDP_ACC.1[25] FMT_MSA.3	Yes Note: In section 6.8 is a rational for why the MSA.3 requirements are not necessary for the TOE defined in this PP.
FDP_ACF.1(2)	FDP_ACC.1[25] FMT_MSA.3	Yes Note: In section 6.8 is a rational for why the MSA.3 requirements are not necessary for the TOE defined in this PP.
FDP_IFC.1	FDP_IFF.1-NIAP-0407	Yes
FDP_IFC.2(1)	FDP_IFF.1-NIAP-0407	Yes
FDP_IFC.2(2)	FDP_IFF.1-NIAP-0407	Yes
FDP_IFF.1-NIAP-0407(1)	FDP_IFC.1[26] FMT_MSA.3	Yes Note: In section 6.8 is a rational for why the MSA.3 requirements are not necessary for the TOE defined in this PP.

[25] FDP_ACC.2 is hierarchical to FDP_ACC.1 thus satisfying this dependency.
[26] FDP_IFC.2 is hierarchical to FDP_IFC.1 thus satisfying this dependency.

Requirement	Dependency	Satisfied
FDP_IFF.1-NIAP-0407(2)	FDP_IFC.1[26] FMT_MSA.3	Yes Note: In section 6.8 is a rationale for why the MSA.3 requirements are not necessary for the TOE defined in this PP.
FDP_IFF.1-NIAP-0407(3)	FDP_IFC.1[26] FMT_MSA.3	Yes Note: In section 6.8 is a rationale for why the MSA.3 requirements are not necessary for the TOE defined in this PP.
FDP_RIP.2	None	N/A
FDP_UCT.1	FTP_ITC.1 or FTP_TRP.1 FDP_ACC.1 or FDP_IFC.1	Yes Yes
FIA_AFL.1	FIA_UAU.1[27]	Yes
FIA_ATD.1	None	N/A
FIA_UAU.2	FIA_UID.1[28]	Yes
FIA_UID.2	None	N/A
FIA_USB.1-NIAP-0415	FIA_ATD.1	Yes
FMT_MOF.1(1)	FMT_SMF.1 FMT_SMR.1[29]	Yes
FMT_MOF.1(2)	FMT_SMF.1 FMT_SMR.1[29]	Yes
FMT_MOF.1(3)	FMT_SMF.1 FMT_SMR.1[29]	Yes

[27] FIA_UAU.2 is hierarchical to FIA_UAU.1 thus satisfying this dependency.

[28] FIA_UID.2 is hierarchical to FIA_UID.1 thus satisfying this dependency.

[29] FMT_SMR.2 is hierarchical to FMT_SMR.1 thus satisfying this dependency.

Requirement	Dependency	Satisfied
FMT_MTD.1	FMT_SMF.1 FMT_SMR.1[29]	Yes
FMT_MTD.2	FMT_MDT.1 FMT_SMR.1[29]	Yes
FMT_SMF.1	None	N/A
FMT_SMR.2	FIA_UID.1[28]	Yes
FMT_SMR.3	FMT_SMR.1[29]	Yes
FPT_RCV.2	AGD_ADM.1 ADV_SPM.1	Yes
FPT_RPL.1	None	N/A
FPT_RVM.1	None	N/A
FPT_SEP.2	None	N/A
FPT_STM.1	None	N/A
FPT_TST_(EXP).4	None	N/A
FPT_TST_(EXP).5	None	N/A
FRU_RSA.1	None	N/A
FTA_SSL.1	FIA_UAU.1[27]	Yes
FTA_SSL.2	FIA_UAU.1[27]	Yes
FTA_TAB.1	None	N/A
FTP_TRP.1	None	N/A

6.6.1 Cellular Communications Package for Satisfying all Dependencies Rationale

Table 19 Functional Requirement Dependencies for Cellular Communications

Requirement	Dependency	Satisfied
FDP_IFC.1(A)	FDP_IFF.1-NIAP-0407(A)	Yes
FDP_IFC.1(B)	FDP_IFF.1-NIAP-0407(B)	Yes
FDP_IFF.1-NIAP-0407(A)	FDP_IFC.1(A) FMT_MSA.3	Yes Note: In section 6.8 is a rationale for why MSA.3 requirements are not necessary for the TOE defined in this PP.
FDP_IFF.1-NIAP-0407(B)	FDP_IFC.1(B) FMT_MSA.3	Yes Note: In section 6.8 is a rationale for why MSA.3 requirements are not necessary for the TOE defined in this PP.

6.7 Rationale for Explicit Requirements

141 Table 20 presents the rationale for the inclusion of the explicit functional and assurance requirements found in this PP. The explicit requirements that are included as NIAP interpretations do not require a rationale for their inclusion per CCEVS management.

Table 20 Rationale for Explicit Requirements

Explicit Requirement	Identifier	Rationale

Explicit Requirement	Identifier	Rationale
FCS_BCM_(EXP).1	Baseline cryptographic module	The CC does not provide a means of specifying a cryptographic module baseline for implementations developed in hardware, in software, or in hardware/software combinations. FCS_BCM_(EXP).1 provides for the specification of the required FIPS certification based on the implementation.
FCS_CKM_(EXP).1	Cryptographic key validation and packaging	The CC cryptographic support section does not specifically address the concepts of key validation techniques and key packaging. Although closely tied to generated keys, these concepts typically get implemented after, not during, the actual generation of a key. In this PP, FCS_CKM_EXP.1 allows for specifically addressing these key management-related concepts.
FCS_CKM_(EXP).2	Cryptographic key handling and storage	The CC does not provide components for key handling and storage. Key access and key destruction components do not address keys being transferred within the device nor key archiving when key is not in use. FCS_CKM_EXP.2 addresses internal key transfer and archiving. It also addresses the handling of storage areas where keys reside.

Explicit Requirement	Identifier	Rationale
FCS_COA_(EXP).1	Cryptographic operations availability	The CC FCS families address the management of cryptographic keys and the operational use of those cryptographic keys to help satisfy several high-level security objectives. Another reason for having the cryptographic functionality in the TOE is for applications to be able to utilize the cryptographic operations. FCS_COA_EXP.1 was created to require a means for applications to be able to utilize the cryptographic functionality contained in the TOE.

Explicit Requirement	Identifier	Rationale
FCS_COP_(EXP).1	Random number generation	The CC cryptographic operation components are focused on specific algorithm types and operations requiring specific key sizes. The generation of random numbers can be better stated as an explicit component. Neither algorithms nor keys are required to generate random numbers. Random number generators can use any combination of software-based or hardware-based inputs as long as the RNG/PRNG design requirements are met and the required RNG/PRNG tests are successful.
FPT_TST_(EXP).4	TSF testing (with cryptographic integrity verification)	This explicit requirement is necessary to capture the notion of the TOE using cryptography to verify the integrity of the TSF software. Additionally, the TSF data set that is subject to these tests was reduced to address the notion that it does not make sense to test the integrity of some TSF data and this explicit requirement address that.

Explicit Requirement	Identifier	Rationale
FPT_TST_(EXP).5	Cryptographic self-test	The PP authors felt that the TSF self tests did not adequately address the notion of testing certain aspects of the TSF upon the completion of an operation. This explicit requirement is necessary to capture the notion of the TOE having the ability to test the cryptographic components immediately after the generation of a key. The CC does not contain a requirement that addresses this notion.
ADV_ARC_(EXP).1	Architectural design with justification	These explicit assurance requirements were deemed necessary by NSA to reduce the ambiguity in the associated CC assurance families and to provide the level of assurance appropriate for medium robustness environments.
ADV_FSP_(EXP).1	Security-enforcing high-level design	
ADV_INT_(EXP).1	Modular decomposition	
ADV_LLD_(EXP).1	Security-enforcing low-level design	
AVA_CCA_(EXP).2	Systematic cryptographic module covert channel analysis	

6.8 Rationale for Not Addressing Consistency Instructions

142 This Protection Profile conforms to the Medium Robustness Consistency Guidance except for the following instructions.

- This rationale is to rationalize why the TOE specified in this PP does not need the FMT_MSA.1 and FMT_MSA.3 functional components. The FMT_MSA family allows authorised users control over the management of security attributes. The TOE specified in this PP is a single user device, meaning that for the TOE to be in the evaluated configuration it may only have a single user account along with an administrator account/role. Only one user is to be logged into and using the TOE at any time. The TOE, once installed, generated, and started (IGS), is not meant to have its security attributes managed while being used by the designated user. The security attributes are set during IGS and it is not necessary for the TOE to be able to manage them during normal TOE operation.

- Instruction 20 was modified to allow the administrator or the user acknowledge the security alarm. Also, the requirement to have an audible alarm sound until the administrator acknowledges it, has been removed. Since the PED is a single user, portable device, it is not practical to require the administrator acknowledge an alarm. It is very likely that no administrator will be immediately available while the device is in the possession of the user. When an alert is displayed, the user will acknowledge it, and return the unit to an administrator for analysis at the next opportunity.

- Instruction 24 was not met because FIA_AFL.1-NIAP-0425 is no longer an active interp, CCIMB 111 was used instead. The wording was also altered to reflect that fact that this TOE is a single user device so locking out the user would not be an effective means of security.

- Instruction 27 was not met because FPT_RCV.2-NAIP-0406 is no longer active Interp, CCIMB 056 was used instead.

- Instruction 25 was not met because FPT_USB.1-NIAP-0415 has been superceded by CCIMB 137.

- This PP changed the definition for O.ADMIN_ROLE because it is not plausible for a PED to have remote administration.

- This PP changed the definition for P.ADMIN_ACCESS because it is not plausible for a PED to have remote administration.

- Instruction 30 was not met because the PED has one user with administrative access. Since the single user could assume any of the specified roles (security, cryptographic, auditor, etc) there is no added benefit from the separation of roles.

7 REFERENCES

[1] *Common Criteria for Information Technology Security Evaluation*, CCIMB-99-031, Version 2.1, August 1999.

[2] *Consistency Instruction Manual for development of US Government Protection Profiles For use in Medium Robustness Environments,* Version 2.0, March 1, 2004.

8 GLOSSARY

Access **–** Interaction between an entity and an object that results in the flow or modification of data.

Access Control **–** Security service that controls the use of resources[30] and the disclosure and modification of data.[31]

Accountability **–** Property that allows activities in an IT system to be traced to the entity responsible for the activity.

Administrator **–** A user who has been specifically granted the authority to manage some portion or all of the TOE and whose actions may affect the TSP. Administrators may possess special privileges that provide capabilities to override portions of the TSP.

Assurance **–** A measure of confidence that the security features of an IT system are sufficient to enforce its' security policy.

Asymmetric Cryptographic System **–** A system involving two related transformations; one determined by a public key (the public transformation), and another determined by a private key (the private transformation) with the property that it is computationally infeasible to determine the private transformation (or the private key) from knowledge of the public transformation (and the public key).

Asymmetric Key **–** The corresponding public/private key pair needed to determine the behavior of the public/private transformations that comprise an asymmetric cryptographic system

Attack **–** An intentional act attempting to violate the security policy of an IT system.

Authentication **–** Security measure that verifies a claimed identity.

Authentication data **–** Information used to verify a claimed identity.

Authorization **–** Permission, granted by an entity authorized to do so, to perform functions and access data.

Authorized user **–** An authenticated user who may, in accordance with the TSP, perform an operation.

Availability **–** Timely[32], reliable access to IT resources.

[30] Hardware and software.

[31] Stored or communicated.

Compromise **–** Violation of a security policy.

Confidentiality – A security policy pertaining to disclosure of data.

Critical Security Parameters (CSP) **–** Security-related information (e.g., cryptographic keys, authentication data such as passwords and pins, and cryptographic seeds) appearing in plaintext or otherwise unprotected form and whose disclosure or modification can compromise the security of a cryptographic module or the security of the information protected by the module.

Cryptographic boundary **–** An explicitly defined contiguous perimeter that establishes the physical bounds (for hardware) or logical bounds (for software) of a cryptographic module.

Cryptographic key (key) **–** A parameter used in conjunction with a cryptographic algorithm that determines:

- the transformation of plaintext data into ciphertext data,
- the transformation of ciphertext data into plaintext data,
- a digital signature computed from data,
- the verification of a digital signature computed from data, or
- a digital authentication code computed from data.

Cryptographic Module **–** The set of hardware, software, firmware, or some combination thereof that implements cryptographic logic or processes, including cryptographic algorithms, and is contained within the cryptographic boundary of the module.

Cryptographic Module Security Policy **–** A precise specification of the security rules under which a cryptographic module must operate, including the rules derived from the requirements of this PP and additional rules imposed by the vendor.

Defense-in-Depth (DID) **–** A security design strategy whereby layers of protection are utilized to establish an adequate security posture for an IT system.

Discretionary Access Control (DAC) **–** A means of restricting access to objects based on the identity of subjects and/or groups to which they belong. Those controls are discretionary in the sense that a subject with a certain access permission is capable of passing that permission (perhaps indirectly) on to any other subject.

Embedded Cryptographic Module **–** On that is built as an integral part of a larger and more general surrounding system (i.e., one that is not easily removable from the surrounding system).

[32] According to a defined metric.

Enclave **–** A collection of entities under the control of a single authority and having a homogeneous security policy. They may be logical, or may be based on physical location and proximity.

Entity **–** A subject, object, user or another IT device, which interacts with TOE objects, data, or resources.

External IT entity **–** Any trusted Information Technology (IT) product or system, outside of the TOE, which may, in accordance with the TSP, perform an operation.

Identity – A representation (e.g., a string) uniquely identifying an authorized user, which can either be the full or abbreviated name of that user or a pseudonym.

Integrity – A security policy pertaining to the corruption of data and TSF mechanisms.

Integrity label **–** A security attribute that represents the integrity level of a subject or an object. Integrity labels are used by the OTE as the basis for mandatory integrity control decisions.

Integrity level **–** The combination of a hierarchical level and an optional set of non-hierarchical categories that represent the integrity of data.

Mandatory Access Control (MAC) **–** A means of restricting access to objects based on subject and object sensitivity labels.[33]

Mandatory Integrity Control (MIC) – A means of restricting access to objects based on subject and object integrity labels.

Multilevel **–** The ability to simultaneously handle (e.g., share, process) multiple levels of data, while allowing users at different sensitivity levels to access the system concurrently. The system permits each user to access only the data to which they are authorized access.

Named Object **–** An object that exhibits all of the following characteristics:

- The object may be used to transfer information between subjects of differing user identities within the TSF.
- Subjects in the TOE must be able to requires a specific instance of the object.
- The name used to refer to a specific instance of the object must exist in a context that potentially allows subjects with different user identities to requires the same instance of the object.

[33] The Bell LaPadula model is an example of Mandatory Access Control.

Non-Repudiation – A security policy pertaining to providing one or more of the following:

- To the sender of data, proof of delivery to the intended recipient,
- To the recipient of data, proof of the identity of the user who sent the data.

Object – An entity within the TSC that contains or receives information and upon which subjects perform operations.

Operating Environment – The total environment in which a TOE operates. It includes the physical facility and any physical, procedural, administrative and personnel controls.

Operating System (OS) – An entity within the TSC that causes operations to be performed. Subjects can come in two forms: trusted and untrusted. Trusted subjects are exempt from part or all of the TOE security policies. Untrusted subjects are bound by all TOE security policies.

Operational key – Key intended for protection of operational information or for the production or secure electrical transmissions of key streams

Peer TOEs – Mutually authenticated TOEs that interact to enforce a common security policy.

Public Object – An object for which the TSF unconditionally permits all entities "read" access. Only the TSF or authorized administrators may create, delete, or modify the public objects.

Robustness – A characterization of the strength of a security function, mechanism, service or solution, and the assurance (or confidence) that it is implemented and functioning correctly. DoD has three levels of robustness:

Basic: Security services and mechanisms that equate to good commercial practices.

Medium: Security services and mechanisms that provide for layering of additional safeguards above good commercial practices.

High: Security services and mechanisms that provide the most stringent protection and rigorous security countermeasures.

Secure State – Condition in which all TOE security policies are enforced.

Security attributes – TSF data associated with subjects, objects, and users that are used for the enforcement of the TSP.

***Security level* –** The combination of a hierarchical classification and a set of non-hierarchical categories that represent the sensitivity of the information.

***Sensitivity label* –** A security attribute that represents the security level of an object and that describes the sensitivity (e.g., Classification) of the data in the object. Sensitivity labels are used by the TOE as the basis for mandatory access control decision.

***Split key* –** A variable that consists of two or more components that must be combined to form the operation key variable. The combining process excludes concatenation or interleaving of component variables.

***Subject* –** An entity within the TSC that causes operation to be performed.

***Symmetric key* –** A single, secret key used for both encryption and decryption in symmetric cryptographic algorithms.

***Threat* –** Capabilities, intentions and attack methods of adversaries, or any circumstance or event, with the potential to violate the TOE security policy.

***Threat Agent* –** Any human user or Information Technology (IT) product or system, which may attempt to violate the TSP and perform an unauthorized operation with the TOE.

***User* –** Any entity (human user or external IT entity) outside the TOE that interacts with the TOE.

***Vulnerability* –** A weakness that can be exploited to violate the TOE security policy.

9 ACRONYMS

AP	Access Point
CC	Common Criteria
CCIMB	Common Criteria Interpretations Management Board
CM	Configuration Management
CSP	Cryptographic security parameter
DoD	Department of Defense
EAL	Evaluation Assurance Level
IATF	Information Assurance Technical Framework
IT	Information Technology
MAC	Mandatory Access Control
NIAP	National Information Assurance Partnership
NIST	National Institute of Standards Technology
NSA	National Security Agency
PKI	Public Key Infrastructure
PP	Protection Profile
SBU	Sensitive But Unclassified
SCIF	Sensitive Compartmented Information Facilities
SFP	Security Functional Policies
SFR	Security Functional Requirement
SOF	Strength of Function
ST	Security Target
TOE	Target of Evaluation
TSC	TOE Scope of Control

TSE	TOE Security Environment
TSF	TOE Security Functions
TSFI	TSF interfaces
TSP	TOE Security Policy
TTAP/CCEVS	Trust Technology Assessment Program/ Common Criteria Evaluation Standard Scheme

10 ROBUSTNESS ENVIRONMENT CHARACTERIZATION

10.1 General Environmental Characterization

143 In trying to specify the environments in which TOEs with various levels of robustness are appropriate, it is useful to first discuss the two defining factors that characterize that environment: value of the resources and authorization of the entities to those resources.

144 In general terms, the environment for a TOE can be characterized by the authorization (or lack of authorization) the least trustworthy entity has with respect to the highest value of TOE resources (i.e. the TOE itself and all of the data processed by the TOE).

145 Note that there are an infinite number of combinations of entity authorization and value of resources; this conceptually "makes sense" because there are an infinite number of potential environments, depending on how the resources are valued by the organization, and the variety of authorizations the organization defines for the associated entities. In the next section, these two environmental factors will be related to the robustness required for selection of an appropriate TOE.

10.1.1 Value of Resources

146 Value of the resources associated with the TOE includes the data being processed or used by the TOE, as well as the TOE itself (for example, a real-time control processor). "Value" is assigned by the using organization. For example, in the DoD low-value data might be equivalent to data marked "FOUO", while high-value data may be those classified Top Secret. In a commercial enterprise, low-value data might be the internal organizational structure as captured in the corporate on-line phone book, while high-value data might be corporate research results for the next generation product. Note that when considering the value of the data one must also consider the value of data or resources that are accessible through exploitation of the TOE. For example, a firewall may have "low value" data itself, but it might protect an enclave with high value data. If the firewall was being depended upon to protect the high value data, then it must be treated as a high-value-data TOE.

10.1.2 Authorization of Entities

147 Authorization that entities (users, administrators, other IT systems) have with respect to the TOE (and thus the resources of that TOE, including the TOE itself) is an abstract concept reflecting a combination of the trustworthiness of an entity and the access and privileges granted to that entity with respect to the resources of the

TOE. For instance, entities that have total authorization to all data on the TOE are at one end of this spectrum; these entities may have privileges that allow them to read, write, and modify anything on the TOE, including all TSF data. Entities at the other end of the spectrum are those that are authorized to few or no TOE resources. For example, in the case of a router, non-administrative entities may have their packets routed by the TOE, but that is the extent of their authorization to the TOE's resources. In the case of an OS, an entity may not be allowed to log on to the TOE at all (that is, they are not valid users listed in the OS's user database).

148 It is important to note that authorization **does not** refer to the **access** that the entities actually have to the TOE or its data. For example, suppose the owner of the system determines that no one other than employees was authorized to certain data on a TOE, yet they connect the TOE to the Internet. There are millions of entities that are not **authorized** to the data (because they are not employees), but they actually have connectivity to the TOE through the Internet and thus can attempt to access the TOE and its associated resources.

149 Entities are characterized according to the value of resources to which they are authorized; the extent of their authorization is implicitly a measure of how trustworthy the entity is with respect to compromise of the data (that is, compromise of any of the applicable security policies; e.g., confidentiality, integrity, availability). In other words, in this model the greater the extent of an entity's authorization, the more trustworthy (with respect to applicable policies) that entity is.

10.1.3 Selection of Appropriate Robustness Levels

150 Robustness is a characteristic of a TOE defining how well it can protect itself and its resources; a more robust TOE is better able to protect itself. This section relates the defining factors of IT environments, authorization, and value of resources to the selection of appropriate robustness levels.

151 When assessing any environment with respect to Information Assurance the critical point to consider is the likelihood of an attempted security policy compromise, which was characterized in the previous section in terms of entity authorization and resource value. As previously mentioned, robustness is a characteristic of a TOE that reflects the extent to which a TOE can protect itself and its resources. It follows that as the likelihood of an attempted resource compromise increases, the robustness of an appropriate TOE should also increase.

152 It is critical to note that several combinations of the environmental factors will result in environments in which the likelihood of an attempted security policy compromise is similar. Consider the following two cases:

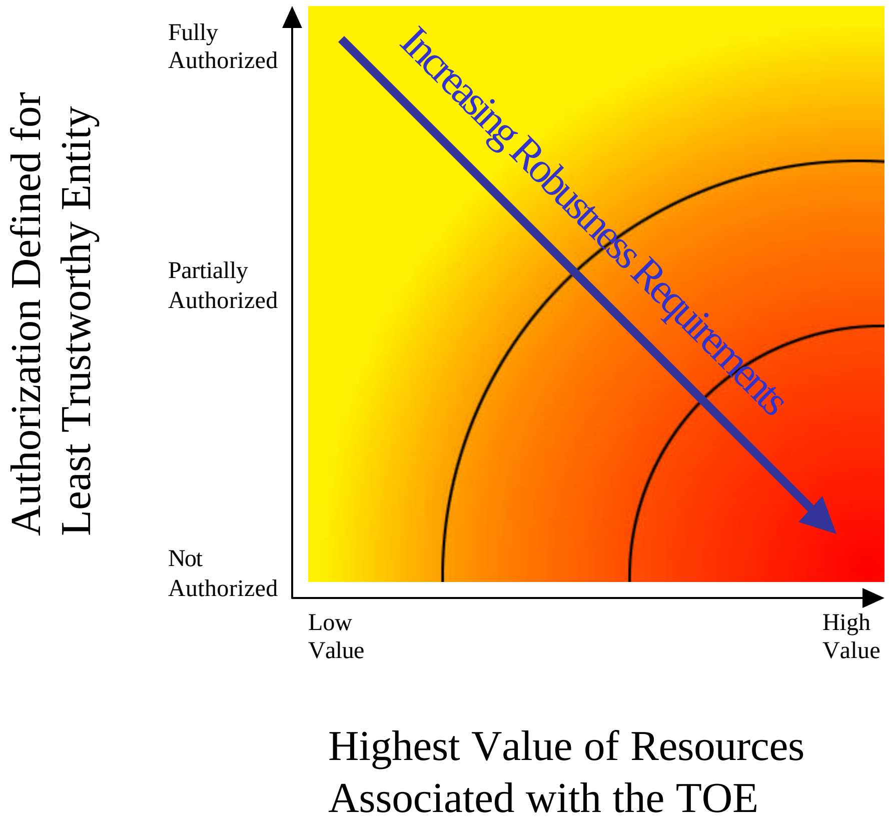
153 The first case is a TOE that processes only low-value data. Although the organization has stated that only its employees are authorized to log on to the

system and access the data, the system is connected to the Internet to allow authorized employees to access the system from home. In this case, the least trusted entities would be unauthorized entities (e.g. non-employees) exposed to the TOE because of the Internet connectivity. However, since only low-value data are being processed, the likelihood that unauthorized entities would find it worth their while to attempt to compromise the data on the system is low and selection of a basic robustness TOE would be appropriate.

154 The second case is a TOE that processes high-value (e.g., classified) information. The organization requires that the TOE be stand-alone, and that every user with physical and logical access to the TOE undergo an investigation so that they are authorized to the highest value data on the TOE. Because of the extensive checks done during this investigation, the organization is assured that only highly trusted users are authorized to use the TOE. In this case, even though high value information is being processed, it is unlikely that a compromise of that data will be attempted because of the authorization and trustworthiness of the users and once again, selection of a basic robustness TOE would be appropriate.

155 The preceding examples demonstrated that it is possible for radically different combinations of entity authorization/resource values to result in a similar likelihood of an attempted compromise. As mentioned earlier, the robustness of a system is an indication of the protection being provided to counter compromise attempts. Therefore, a basic robustness system should be sufficient to counter compromise attempts where the likelihood of an attempted compromise is low. The following chart depicts the "universe" of environments characterized by the two factors discussed in the previous section: on one axis is the authorization defined for the least trustworthy entity, and on the other axis is the highest value of resources associated with the TOE.

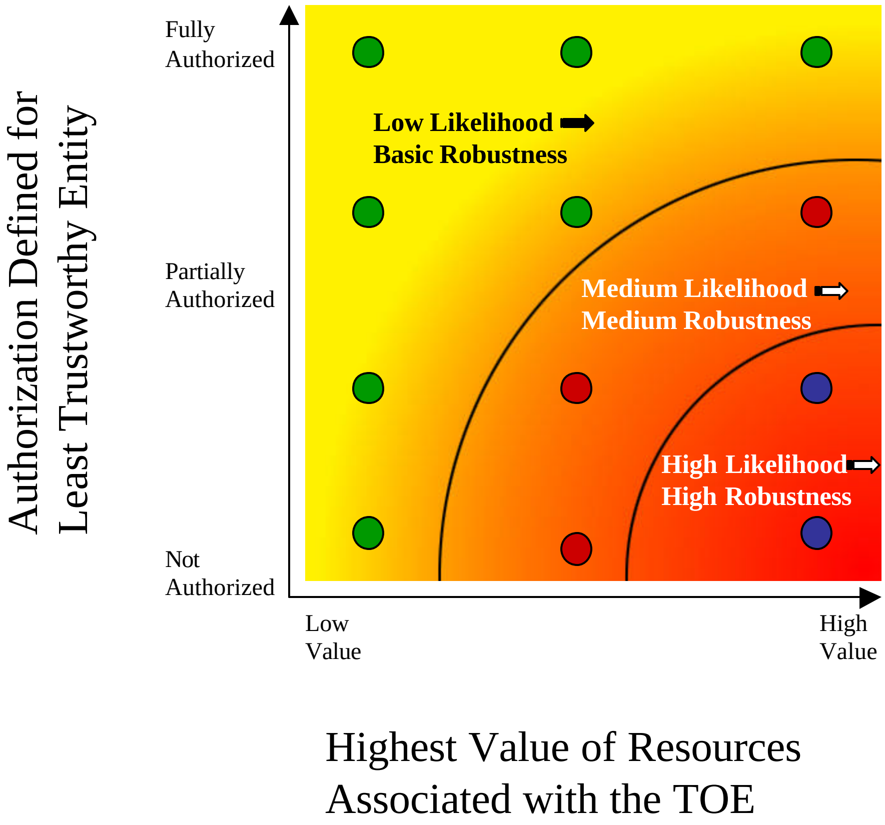
156 As depicted in the following figure, the robustness of the TOEs required in each environment steadily increases as one goes from the upper left of the chart to the lower right; this corresponds to the need to counter increasingly likely attack attempts by the least trustworthy entities in the environment. Note that the shading of the chart is intended to reflect- the notion that different environments engender similar levels of "likelihood of attempted compromise", signified by a similar color. Further, the delineations between such environments are not stark, but rather are finely grained and gradual.



157 While it would be possible to create many different "levels of robustness" at small intervals along the “Increasing Robustness Requirements” line to counter the increasing likelihood of attempted compromise due to those attacks, it would not be practical nor particularly useful. Instead, in order to implement the robustness strategy where there are only three robustness levels: Basic, Medium, and High, the graph is divided into three sections, with each section corresponding to a set of environments where the likelihood of attempted compromise is roughly similar. This is graphically depicted in the following chart.

158 In this second representation of environments and the robustness plane below, the “dots” represent given instantiations of environments; like-colored dots define environments with a similar likelihood of attempted compromise. Correspondingly, a TOE with a given robustness should provide sufficient protection for environments characterized by like-colored dots. In choosing the appropriateness of a given robustness level TOE PP for an environment, then, the user must first consider the lowest authorization for an entity as well as the highest value of the resources in that environment. This should result in a “point” in the chart above, corresponding to the likelihood that that entity will attempt to compromise the most valuable resource in the environment. The appropriate robustness level for the specified TOE to counter this likelihood can then be chosen.

159 The difficult part of this activity is differentiating the authorization of various entities, as well as determining the relative values of resources; (e.g., what constitutes "low value" data vs. "medium value" data). Because every organization will be different, a rigorous definition is not possible. In Section 3 of this PP, the targeted threat level for a medium robustness TOE is characterized. This information is provided to help organizations using this PP -ensure that the functional requirements specified by this medium robustness PP are appropriate for their intended application of a compliant TOE.



11 EXPLANATORY MATERIAL FOR EXPLICIT ASSURANCE REQUIREMENTS

11.1 ADV_INT_(EXP).1

160 This explicit component was created to levy different modularity metrics on the SFP-enforcing modules and non-SFP-enforcing modules.

161 The parts of the TSF that implement an SFP (in this component, SFP-enforcing is used to designate modules that enforce an SFP) that is determined and assigned by the PP/ST author, are those modules that interact (defined in the coupling analysis) with the module or modules that provide the TSFI for that SFP with justified exceptions. The intent is that all of the modules that play an SFR related role (as opposed to modules that provide infrastructure support, such as scheduling, reading binary data from the disk) in enforcing an SFP are identified as SFP-enforcing. The remaining modules in the TSF are deemed non-SFP-enforcing modules, since they could be TSP-enforcing (e.g., enforcing a policy not assigned to this component), as well as TSP-supporting.

11.1.1 Objectives

162 This component addresses the internal structure of the software TSF. The SFP-enforcing modules require stricter adherence to the coupling and cohesion metrics than the metrics levied on the non-SFP-enforcing modules due to their key role in policy enforcement. While the non-SFP-enforcing modules also play a role in enforcing policy, their role is not as critical as the SFP-enforcing modules, therefore, the degree of coupling and cohesion required of these modules is not as restrictive. It is expected that all of the TSF modules are designed using good software engineering practice, whether they are developed by the developer or incorporated as a third party implementation into the TSF.

163 Requirements are presented for modular decomposition of the SFP-enforcing and non-SFP-enforcing functionality within the TSF. These requirements, when applied to the internal structure of the TSF, should result in improvements that aid both the developer and the evaluator in understanding the TSF, and also provides the basis for designing and evaluating test suites. Further, improving understandability of the TSF should assist the developer in simplifying its maintainability. The principal goal achieved by inclusion of the requirements from the ADV_INT class in a PP/ST is understandability of the TSF.

164 Modular design aids in achieving understandability by clarifying what dependencies and interactions a module has on other modules (*coupling*), by including in a

module only tasks that are strongly related to each other (*cohesion*), and by illuminating the design of a module by using internal structuring and reduced complexity. The use of modular design reduces the interdependence between elements of the TSF and thus reduces the risk that a change or error in one module will have effects throughout the TOE. Its use enhances clarity of design and provides for increased assurance that unexpected effects do not occur. Additional desirable properties of modular decomposition are a reduction in the amount of redundant or unneeded code.

165 The incorporation of modular decomposition into the design and implementation process must be accompanied by sound software engineering considerations. A practical, useful software system will usually entail some undesirable coupling among modules, some modules that include loosely-related functions, and some subtlety or complexity in a module's design. These deviations from the ideals of modular decomposition are often deemed necessary to achieve some goal or constraint, be it related to performance, compatibility, future planned functionality, or some other factors, and may be acceptable, based on the developer's justification for them. In applying the requirements of this class, due consideration must be given to sound software engineering principles; however, the overall objective of achieving understandability must be achieved.

166 Another key component to reducing complexity is the use of coding standards. Coding standards are used as a reference to ensure programmers generate code that can be easily understood by individuals (e.g., code maintainers, code reviewers, evaluators) that are not intimately familiar with the nuances of the functions performed by the code. For example, coding standards ensure that meaningful names are given to variables and data structures, the code has a structure that is similar to code developed by other programmers, loops used in the code are understandable (e.g., leaving a loop to another section of code and returning is undesirable), the use of pointers to variables/data structures is straightforward, and the code is suitably commented (inline and/or by a preamble). The use of coding standards helps to eliminate errors in code development and maintenance, and assists the development team in performing code walk-throughs. Some aspects of coding standards are specific to a given program language (e.g., the C language may have a different standard than the Java language or assembly level code). It is expected that the coding standards are appropriately followed for the employed programming language(s). The requirements in this component allow for exceptions to the adherence of coding standards that may be necessary for reasons of performance, or some other factors, but these deviations must be justified (on a per module basis) as to why they are necessary. Any justification provided must address why the deviation does not unduly introduce complexity into the module, since ultimately, the goal of adhering to coding standards is to improve clarity.

167 Design complexity minimization is a key characteristic of a reference validation mechanism, the purpose of which is to arrive at a TSF that is easily understood so that it can be completely analyzed. (There are other important characteristics of a reference validation mechanism, such as TSF self-protection and TSP non-

bypassability; these other characteristics are covered by requirements from other classes.)

11.1.2 Application Notes

168 Several of the elements within this component refer to the architectural description. The architectural description is at a similar level of abstraction as the low-level design, in that it is concerned with the modules of the TSF. Whereas the low-level design describes the design of the modules of the TSF, the purpose of the architectural description is to provide evidence of modular decomposition of the TSF. Both the low-level design and the implementation representation are required to be in compliance with the architectural description, to provide assurance that these TSF representations possess the required modular decomposition.

169 This component requires the PP or ST author to fill in an assignment with the SFPs that are felt to be critical to the TOE and therefore their resulting design and implementation require stricter metrics for modularity. The SFPs can be those explicitly identified in the CC (i.e., FDP_ACC, FDP_IFF) by simply placing the appropriate label as specified in those requirements, or other policies determined by the PP/ST author (e.g., I&A, Audit), in which case, the PP/ST author should explicitly identify all of the SFRs that they intend to satisfy a policy that is not explicitly stated in the CC. This is necessary since currently a convention does not exist to place a convenient label on these policies.

170 The requirements in this component refer to SFP-enforcing and non-SFP-enforcing portions of the TSF. The non-SFP-enforcing portions of the TSF consist of the TSP-supporting modules and TSP-enforcing modules that do not play a role in the enforcement of the SFP(s) identified in ADV_INT_(EXP).1.4D as depicted in the Figure E1, where in this example, non-SFP-enforcing is everything in the TSF other than the SFP-enforcing functions.

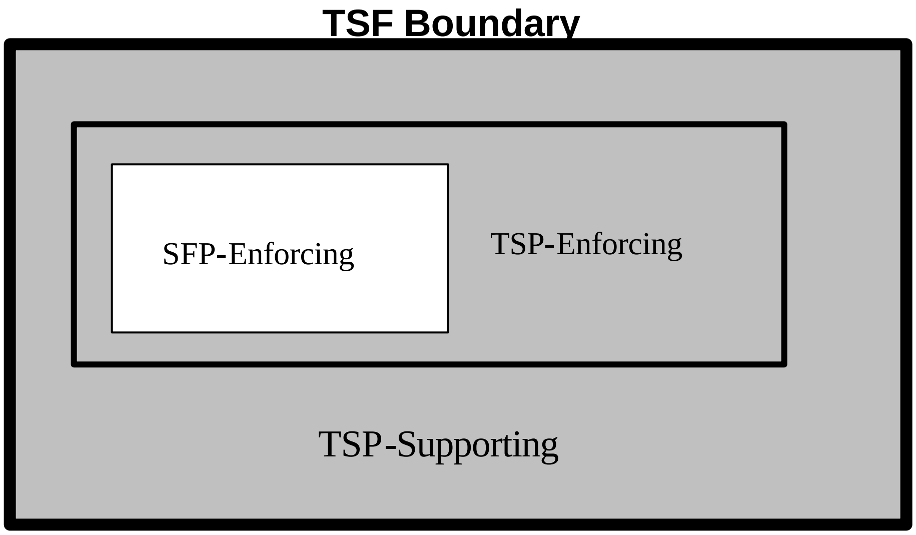


Figure E1. SFP-enforcing may only be a subset of TSP-enforcing functions.

171 The developer is required to identify the modules that are SFP-enforcing and implicitly the remaining modules, which will be non-SFP-enforcing. As stated earlier, the SFP-enforcing modules are those modules that interact with the module or modules that provide the TSFI for that SFP with justified exceptions. The justification of the non-SFP-enforcing modules (ADV_INT_(EXP).1.3C) is required only for those modules that interact with SFP-enforcing modules and not for all non-SFP-enforcing modules. As depicted in the Figure E2 below, if a TSFI has already been designated as non-SFP-enforcing then the designation of the modules interacting with the module providing the TSFI do not have to be justified (e.g., modules X, Y, Z). The justification of the designation is only necessary for the module(s) that interact with a module that provides a TSFI that is SFP-enforcing (e.g., modules D, E, F (since it is writing to a global variable that Module A is reading, but in this example, it is not an SFP-enforcing variable).

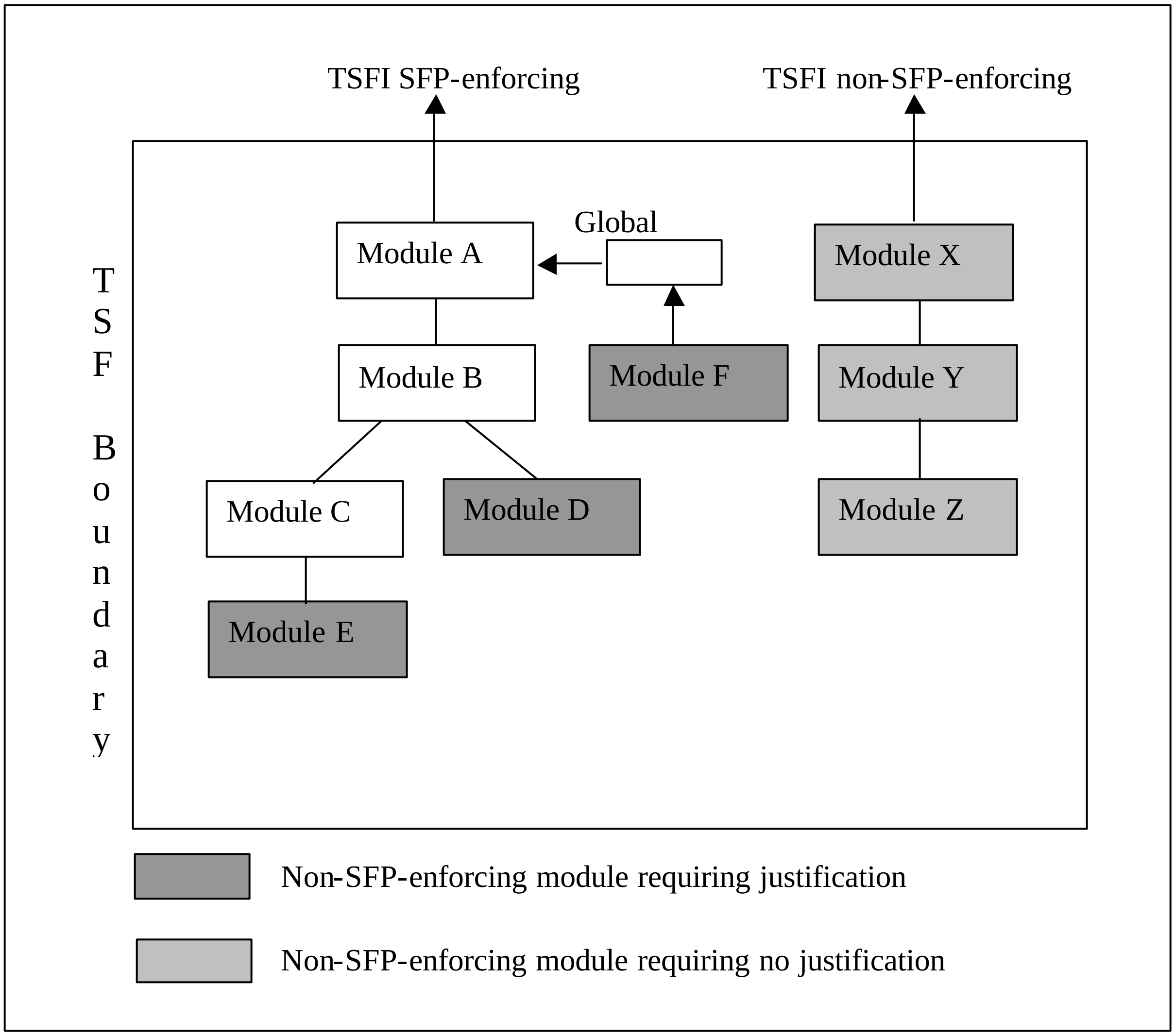


Figure E2. Example of non-SFP-enforcing modules requiring justification.

172 The modules identified in the architectural description are the same as the modules identified in the low-level design.

11.1.3 Terms, Definitions and Background

173 The following terms are used in the requirements for software internal structuring. Some of these are derived from the Institute of Electrical and Electronics Engineers *Glossary of software engineering terminology, IEEE Std 610.12-1990.*

Module – One or more source code files that cannot be decomposed into smaller compliable units.

Modular decomposition – The process of breaking a system into components to facilitate design and development.

Cohesion (also called module strength) – The manner and degree to which the tasks performed by a single software module are related to one another; types of cohesion include coincidental, communicational, functional, logical, sequential, and temporal. These types of cohesion are characterized below, listed in order of decreasing desirability.

Functional cohesion – A module with this characteristic performs activities related to a single purpose. A functionally cohesive module transforms a single type of input into a single type of output, such as a stack manager or a queue manager.

Sequential cohesion – A module with this characteristic contains functions each of whose output is input for the following function in the module. An example of a sequentially cohesive module is one that contains the functions to write audit records and to maintain a running count of the accumulated number of audit violations of a specified type.

Communicational cohesion – A module with this characteristic contains functions that produce output for, or use output from, other functions within the module. An example of a communicationally cohesive module is an access check module that includes mandatory, discretionary, and capability checks.

Temporal cohesion – A module with this characteristic contains functions that need to be executed at about the same time. Examples of temporally cohesive modules include initialization, recover, and shutdown modules.

Logical (or procedural) cohesion – A module with this characteristic performs similar activities on different data structures. A module exhibits logical cohesion if its functions perform related, but different, operations on different inputs.

Coincidental cohesion – A module with this characteristic performs unrelated, or loosely related activities.

Coupling **–** The manner and degree of interdependence between software modules; types of coupling include call, common and content coupling. These types of coupling are characterized below, listed in the order of decreasing desirability.

Call **–** Two modules are call coupled if they communicate strictly through the use of their documented function calls; examples of call coupling are data, stamp, and control, which are defined below.

Data **–** Two modules are data coupled if they communicate strictly through the use of call parameters that represent single data items.

Stamp **–** Two modules are stamp coupled if they communicate through the use of call parameters that comprise multiple fields or that have meaningful internal structures.

Control **–** Two modules are control coupled if one passes information that is intended to influence the internal logic of the other.

Common **–** Two modules are common coupled if they share a common data area or a common system resource. Global variables indicate that modules using those global variables are common coupled.[34]

Common coupling through global variables is generally allowed, but only to a limited degree. For example, variables that are placed into a global area, but are used by only a single module, are inappropriately placed, and should be removed. Other factors that need to be considered in assessing the suitability of global variables are:

The number of modules that modify a global variable: In general, only a single module should be allocated the responsibility for controlling the contents of a global variable, but there may be situations in which a second module may share that responsibility; in such a case, sufficient justification must be provided. It is unacceptable for this responsibility to be shared by more than two modules. (In making this assessment, care should be given to determining the module actually responsible for the contents of the variable; for example, if a single routine is used to modify the variable, but that routine simply performs the modification requested by its caller, it is the calling module that is responsible, and there may be more than one such module). Further, as part of the complexity determination, if two modules are responsible for the contents of a global variable, there should be clear indications of how the modifications are coordinated between them.

[34] It can be argued that modules sharing definitions, such as data structure definitions, are common coupled. However, for the purposes of this analysis, shared definitions are considered acceptable, but are subject to the cohesion analysis.

The number of modules that reference a global variable: Although there is generally no limit on the number of modules that reference a global variable, cases in which many modules make such a reference should be examined for validity and necessity.

Content – Two modules are content coupled if one can make direct reference to the internals of the other (e.g., modifying code of, or referencing labels internal to, the other module). The result is that some or all of the content of one module are effectively included in the other. Content coupling can be though of as using unadvertised module interfaces; this is in contract to call coupling, which uses only advertised module interfaces.

Call tree – A diagram that identifies the modules in a system and shows which modules call one another. All the modules named in a call tree that originates with (i.e., is rooted by) a specific module are the modules that directly or indirectly implement the functions of the originating module.

Software engineering - The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. As with engineering practices in general, some amount of judgment must be used in applying engineering principles. Many factors affect choices, not just the application of measures of modular decomposition, layering, and minimization. For example, a developer may design a system with future applications in mind that will not be implemented initially. The developer may choose to include some logic to handle these future applications without fully implementing them; further, the developer may include some calls to as-yet unimplemented modules, leaving *call stubs*. The developer's justification for such deviations from well-structured programs will have to be assessed using judgment, as well as the application of good software engineering discipline.

Complexity - This is a measure of how difficult software is to understand, and thus to analyze, test, and maintain. Reducing complexity is the ultimate goal for using modular decomposition, layering and minimization. Controlling coupling and cohesion contributes significantly to this goal.

174 A good deal of effort in the software engineering field has been expended in attempting to develop metrics to measure the complexity of source code. Most of these metrics use easily computed properties of the source code, such as the number of operators and operands, the complexity of the control flow graph (*cyclomatic complexity*), the number of lines of source code, the ratio of comments to executable code, and similar measures. Coding standards have been found to be a useful tool in generating code that is more readily understood.

175 While this component calls for the evaluator to perform a *complexity analysis*, it is expected that the developer will provide support for the claims that the modules are not overly complex (ADV_INT_(EXP).1.3D, ADV_INT_(EXP).1.6D,

ADV_INT_(EXP).1.9C). This support could include the developer's programming standards, and an indication that all modules meet the standard (or that there are some exceptions that are justified by software engineering arguments). It could include the results of tools used to measure some of the properties of the source code. Or it could include other support that the developer finds appropriate.

11.2 ADV_FSP_(EXP).1

176 The functional specification is a description of the user-visible interface to the TSF. It contains an instantiation of the TOE security functional requirements. The functional specification has to completely address all of the user-visible TOE security functional requirements.

11.2.1 Application Notes

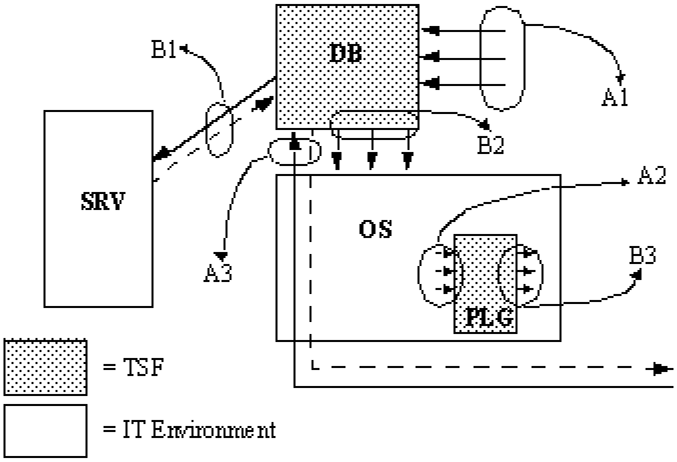
177 A description of the TSF interfaces (TSFI) provides fundamental evidence on which assurance in the TOE can be built. Fundamentally, the functional specification provides a description of *what* the TSF provides to users (as opposed to the high-level design and low-level design, which provide a description of *how* the functionality is provided). Further, the functional specification provides this information in the form of interface (TSFI) documentation.

178 In order to identify the software interfaces to the TSF, the parts of the TOE that make up the TSF must be identified. This identification is formally a part of ADV_HLD_EXP analysis. In this analysis, a portion of the TOE is considered to be in the TSF under two conditions:

1. The software contributes to the satisfaction of security functionality specified by a functional requirement in the ST. This is typically all software that runs in a privileged state of the underlying hardware, as well as software that runs in unprivileged states that performs security functionality.
2. The software used by administrators in order to perform security management activities specified in the guidance documentation. These activities are a superset of those specified by any FMT_* functional requirements in the ST.

179 Identification of the TSFI is a complex undertaking. The TSF is providing services and resources, and so the TSFI are interfaces *to* the security services/resources the TSF is providing. This is especially relevant for TSFs that have dependencies on the IT environment, because not only is the TSF providing security services (and thus exposing TSFI), but it is also *using* services of the IT environment. While these are (using the general term) interfaces between the TSF and the IT environment, they are not TSFI. Nonetheless, it is vital to document their existence to integrators and consumers of the system, and thus documentation requirements for these interfaces are specified in ADV_ING.

180 This concept (and concepts to be discussed in the following paragraphs) is illustrated in the following figure.



181 The figure above illustrates a TOE (a database management system) that has dependencies on the IT environment. The shaded boxes represent the TSF, while the un-shaded boxes represent IT entities in the environment. The TSF comprises the database engine and management GUIs (represented by the box labeled “DB”) and a kernel module that runs as part of the OS that performs some security function (represented by the box labeled “PLG”). The TSF kernel module has entry points defined by the OS specification that the OS will call to invoke some function (this could be a device driver, or an authentication module, etc.). The key is that this pluggable kernel module is providing security services specified by functional requirements in the ST. The IT environment consists of the operating system (represented by the box labeled “OS”) itself, as well as an external server (labeled SRV). This external server, like the OS, provides a service that the TSF depends on, and thus needs to be in the IT environment. Interfaces in the figure are labeled Ax for TSFI, and Bx for interfaces to be documented in AGD_ING. Each of these groups of interfaces is now discussed.

182 Interface group A1 represents the prototypical set of TSFI. These are interfaces used to directly access the database and its security functionality and resources.

183 Interface group A2 represent the TSFI that the OS invokes to obtain the functionality provided by the pluggable module. These are contrasted with interface group B3, which represent calls that the pluggable module makes to obtain services from the IT environment.

184 Interface group A3 represents TSFI that “pass through” the IT environment. In this case, the DBMS communicates over the network using a proprietary application-level protocol. While the IT environment is responsible for providing various supporting protocols (e.g., Ethernet, IP, TCP), the application layer protocol that is used to obtain services from the DBMS is a TSFI and must be documented as such. The dotted line indicates return values/services from the TSF over the network connection.

185 Non-TSFI interfaces pictured are labeled Bx. Interface group B1 is the most complex of these, because the architecture of the system and environmental assumptions and conditions will drive its analysis. In the first case, assume that, either through an environmental assumption or an IT environmental requirement, the network link between the DB and SRV is protected (it could be on a separate subnet, or it could be protected by a firewall such that only the DB could connect to the port on the SRV) such that only the DB has access to the SRV. In this case, the interface needs only to be documented in the integrator guidance, since untrusted users are unable to gain access.

186 However, consider the case where SRV is now just "somewhere on the network", and now the port that the DB opens up to communicate with the SRV is "exposed" to untrusted users. In this case, while the interface presented by the DB (the TSF) still only needs to be documented in the integrator guidance, additional considerations with respect to vulnerabilities may need to be documented as part of the AVA_VLA activity because of this exposure.

187 In the course of performing its functions, the DB will make system calls down to the OS. This is represented by interface group B2. While these calls are not part of the TSFI, they are an interface that needs to be documented in the integrator guidance.

188 Interface group B3, mentioned previously in connection with interface group A2, is similar to interface group B2 in that these are calls made by the TSF to the IT environment to perform services for the TSF.

189 Having discussed the interfaces in general, the types of TSFI are now discussed in more detail. This discussion categorizes the TSFI into the two categories mentioned previously: TSFI to software directly implementing the SFRs, and TSFI used by administrators.

190 TSFI in the first category are varied in their appearance in a TOE. Most commonly interfaces are thought of as those described in terms of Application Programming Interfaces (APIs), such as kernel calls in a Unix-like operating system. However, interfaces also may be described in terms of menu choices, check boxes, and edit boxes in a GUI; parameter files (the *.INI files and the registry for Microsoft Windows systems); and network communication protocols at all levels of the protocol stack.

191 TSFI in the second category are more complex. While there are three cases that need to be considered (discussed below), for all cases there is an "additional" requirement that the functions that an administrator uses to perform their duties—as documented in administrative guidance—also are part of the TSFI and must be documented and shown to work correctly. The individual cases are as follows:

a) The administrative tool used is also accessible to untrusted users, and runs with some "privilege" itself. In this case the TSFI to be described are similar to those in the first category because the tool itself is privileged.

b) The administrative tool uses the privileges of the invoker to perform its tasks. In this case, the interfaces supporting the activities that the administrator is directed to do by the administrative guidance (AGD_ADM, including FMT_* actions) are part of the TSFI. Other interfaces supported by the tool that the administrator is directed not to use (and thus play no role in supporting the TSP), but that are accessible to non-administrators, are not part of the TSFI because there are no privileges associated with their use. Note that this case differs from the previous one in that the tool does not run with privilege, and therefore is not in and of itself interesting from a security point of view. Also note that when FPT_SEP is included in the ST, the executable image of such tools need to be protected so that an untrusted user cannot replace the tool with a “Trojan” tool.

c) The administrative tool is only accessible to administrative users. In this case the TSFI are identified in the same manner as the previous case. Unlike the previous case, however, the evaluator ascertains that an untrusted user is unable to invoke the tool when FPT_SEP is included in the ST.

192 It is also important to note that some TOEs will have interfaces that one might consider part of the TSFI, but environmental factors remove them from consideration (an example is the case of interface group B1 discussed earlier). Most of these examples are for TOEs to which untrusted users have restricted access. For example, consider a firewall that untrusted users only have access to via the network interfaces, and further that the network interfaces available only support packet-passing (no remote administration, no firewall-provided services such as telnet). Further suppose that the firewall had a command-line interface that logged-in administrators could use to administer the system, or they could use a GUI-based tool that essentially translated the GUI-based checkboxes, textboxes, etc., into scripts that invoked the command-line utilities. Finally, suppose that the administrators were directed in the administrative guidance to use the GUI-based tool in administering the firewall. In this case, the command-line interface does not have to be documented because it is inaccessible to untrusted users, and because the administrators are instructed not use it.

193 The term “administrator” above is used in the sense of an entity that has complete trust with respect to all policies implemented by the TSF. There may be entities that are trusted with respect to some policies (e.g., audit) and not to others (e.g., a flow control policy). In these cases, even though the entity may be referred to as an “administrator”, they need to be treated as untrusted users with respect to policies to which they have no administrative access. So, in the previous firewall example, if there was an auditor role that was allowed direct log-on to the firewall machine, the command-line interfaces not related to audit are now part of the TSFI, because they are accessible to a user that is not trusted with respect to the policies the interfaces provide access to. The point is that such interfaces need to be addressed in the same manner as previously discussed.

194 Hardware interfaces exist as well. Functions provided by the BIOS of various devices may be visible through a “wrapper” interface such as the IOCTLs in a Unix operating system. If the TOE is or includes a hardware device (e.g., a network interface card), the bus interface signals, as well as the interface seen at the network port, must be considered “interfaces.” Switches that can change the behavior of the hardware are also part of the interface.

195 As indicated above, an interface exists at the TSF boundary if it can be used (by an administrator; untrusted user; or another TOE) to affect the behavior of the TSF. The requirements in this family apply to all types of TSFI, not just APIs.

196 All TSFI are *security relevant*, but some interfaces (or aspects of interfaces) are more critical and require more analysis than other interfaces. If an interface plays a role in enforcing any security policy on the system, then that interface is *security enforcing*. Such policies are not limited to the access control policies, but also refer to any functionality provided by one of the SFRs contained in the ST (with exceptions for FPT_SEP and FPT_RVM as detailed below). Note that it is possible that an interface may have various effects and exceptions, some of which may be security enforcing and some of which may not.

197 FPT_SEP and FPT_RVM are SFRs that require a different type of analysis from other SFRs. These requirements are architecturally related, and their implementation (or lack thereof) is not easily (or efficiently) testable at the TSFI. From a terminology standpoint, although implementation (and the associated analysis) of FPT_SEP and FPT_RVM is critical to the trustworthiness of the system, these two SFRs will not be considered as SFRs that are applicable when determining the set of security-enforcing TSFIs as defined in the previous paragraph.

198 Interfaces (or parts of an interface) that need only to function correctly in order for the security policies of the system to be preserved are termed *security supporting*. A security supporting interface typically plays a role in supporting the architectural requirements (FPT_SEP or FPT_RVM), meaning that as long as it can be shown that it does not allow the TSF to be compromised or bypassed no further analysis against SFRs is required. In order for an interface to be security supporting it must have *no* security enforcing aspects. In contrast, a security enforcing interface may have security supporting aspects (for example, the ability to set the system clock may be a security enforcing aspect of an interface, but if that same interface is used to display the system date that effect may only be security supporting).

199 A key aspect for the assurance associated with this component is the concept of the evaluator being able to verify that the developer has correctly categorized the security enforcing and security supporting interfaces. The requirements are structured such that the information required for security supporting interfaces is the *minimum* necessary in order for the evaluator to make this determination in an effective manner.

200 For the purposes of the requirements, interfaces are specified (in varying degrees of detail) in terms of their parameters, parameter descriptions, effects, exceptions, and error messages. Additionally, the purpose of each interface, and the way in which the interface is used (both from the point of view of the external stimulus (e.g., the programmer calling the API, the administrator changing a setting in the registry) and the effect on the TSFI that stimulus has) must be specified. This description of method of use must also specify how those administrative interfaces that are unable to be successfully invoked by untrusted users (case "c" mentioned above) are protected.

201 Parameters are explicit inputs to and outputs from an interface that control the behavior of that interface. For examples, parameters are the arguments supplied to an API; the various fields in a packet for a given network protocol; the individual key values in the Windows Registry; the signals across a set of pins on a chip; etc.

202 A parameter description tells what the parameter is in some meaningful way. For instance, the interface "foo(i)" could be described as having "parameter i which is an integer"; this is not an acceptable parameter description. A description such as "parameter i is an integer that indicates the number of users currently logged in to the system." is required.

203 Effects of an interface describe what the interface does. The effects that need to be described in an FSP are those that are visible at any external interface, not necessarily limited to the one being specified. For instance, the sole effect of an API call is not just the error code it returns. Also, depending on the parameters of an interface, there may be many different effects (for instance, an API might have the first parameter be a "subcommand", and the following parameters be specific to that subcommand. The IOCTL API in some Unix systems is an example of such an interface).

204 Exceptions refer to the processing associated with "special checks" that may be performed by an interface. An example would be an interface that has a certain set of effects for all users except the Superuser; this would be an exception to the normal effect of the interface. Use of a privilege for some kind of special effect would also be covered in this topic.

205 Documenting the errors associated with the TSF is not as straightforward as it might appear, and deserves some discussion. A general principle is that errors generated by the TSF that are visible to the user should be documented. These errors can be the direct result of invoking a TSFI (an API call that returns an error); an indirect error that is easily tied to a TSFI (setting a parameter in a configuration that is error-checked when read, returning an immediate notification); or an indirect error that is not easily tied to a TSFI (setting a parameter that, in combination with certain system states, generates an error condition that occurs at a later time. An example might be resource exhaustion of a TSF resource due to setting a parameter to too low of a value).

206 Errors can take many forms, depending on the interface being described. For an API, the interface itself may return an error code; set a global error condition, or set a certain parameter with an error code. For a configuration file, an incorrectly configured parameter may cause an error message to be written to a log file. For a hardware PCI card, an error condition may raise a signal on the bus, or trigger an exception condition to the CPU.

207 For the purposes of the requirements, errors are divided into two categories. The first category includes *direct errors*, which are directly related to a TSFI; examples are API calls and parameter-checking for configuration files. For this category of errors, the functional specification must document all of the errors that can be returned as a result of invoking a security-enforcing aspect of the interface such that a reader should be able to associate an interface with the errors it is capable of generating. The second category includes *indirect errors*, which are errors that are not directly tied to the invocation of a TSFI, but which are reported to the user as a result of processing that occurs in the TSF. It should be noted that while the condition that causes the indirect error can be documented; it is generally much harder to document all the ways in which that condition can occur.[35] Because of the difficulty associated with documenting all of the ways to cause an error, and because of the cost of documenting all indirect errors compared to the benefit of having them documented, indirect errors are not required to be documented.

208 The ADV_FSP_(EXP).1.2E element defines a requirement that the evaluator determines that the functional specification is an accurate and complete instantiation of the TOE security functional requirements. This provides a direct correspondence between the TOE security functional requirements and the functional specification, in addition to the pairwise correspondences required by the ADV_RCR family. Although the evaluator may use the evidence provided in ADV_RCR as an input to making this determination, ADV_RCR cannot be the basis for a positive finding in this area. The requirement for completeness is intended to be relative to the level of abstraction of the functional specification.

[35] This may even be impossible, if the error message is for a condition that the programmer does not expect to occur, but is inserted as part of "defensive programming."

11.3 ADV_HLD_(EXP).1

209 The high-level design of a TOE provides both context for a description of the TSF, and a thorough description of the TSF in terms of major structural units (i.e. subsystems). It relates these units to the functions that they provide. The high-level design requirements are intended to provide assurance that the TOE provides an architecture appropriate to implement the security-enforcing TOE security functional requirements.

210 To provide context for the description of the TSF, the high-level design describes the entire TOE at a high level. From this description the reader should be able to distinguish between the subsystems that are part of the TSF and those that are not. The remainder of the high-level design document then describes the TSF in more detail.

211 The high-level design refines the functional specification into subsystem descriptions. The functional specification provides a description of *what* the TSF does at its interface; the high-level design provides more insight into the TSF by describing *how* the TSF works in order to perform the functions specified at the TSFI. For each subsystem of the TSF, the high-level design identifies the TSFI implemented in the subsystem, describes the purpose of the subsystem and how the implementation of the TSFI (or portions of the TSFI) is designed. The interrelationships of subsystems are also defined in the high-level design. These interrelationships will be represented as data flows, control flows, etc. among the subsystems. It should be noted that this description is at a high level; low-level implementation detail is not necessary at this level of abstraction.

212 The developer is expected to describe the design of the TSF in terms of subsystems. The term “subsystem” is used here to express the idea of decomposing the TSF into a relatively small number of parts. While the developer is not required to actually have “subsystems”, the developer is expected to represent a similar level of decomposition. For example, a design may be similarly decomposed using “layers”, “domains”, or “servers”.

213 A security enforcing subsystem is a subsystem that provides mechanisms for enforcing an element of the TSP, or directly supports a subsystem that is responsible for enforcing the TSP. If a subsystem provides a security-enforcing interface, then the subsystem is security enforcing. If a subsystem does not provide any security enforcing TSFIs, its mechanisms still must preserve the security of the TSF; such subsystems are termed security supporting.

214 As was the case with ADV_FSP_EXP, the set of SFRs that determine the TSP for the purposes of this component do not include FPT_SEP and FPT_RVM. Those two architectural functional requirements require a different type of analysis than that needed for all other SFRs. A security-enforcing subsystem is one that is designed to implement an SFR other than FPT_SEP and FPT_RVM; the design information and

justification for the FPT_SEP and FPT_RVM requirements is given as a result of the ADV_ARC_EXP component.

215 The ADV_HLD_EXP component requires that the developer must identify all subsystems of the TSF (not just the security-enforcing ones). In general, the component requires that the security-enforcing aspects of the subsystems be described in more detail than the security-supporting aspects. The descriptions for the security-enforcing aspects should provide the reader with enough information to determine *how* the implementation of the SFRs is designed, while the description for the security-supporting aspects should provide the reader enough assurance to determine that 1) all security-enforcing behavior has been identified and 2) the subsystems or portions of subsystems that are security supporting have been correctly classified.

216 The ADV_HLD_(EXP).1.2E element for this component defines a requirement that the evaluator determine that the high-level design is an accurate and complete instantiation of the user-visible TOE security functional requirements. This provides a direct correspondence between the TOE security functional requirements and the high-level design, in addition to the pair wise correspondences required by the ADV_RCR family. Although the evaluator may use the evidence provided in ADV_RCR as an input to making this determination, ADV_RCR cannot be the basis for a positive finding in this area. The requirement for completeness is intended to be relative to the level of abstraction of the high-level design. Note that for this element FPT_SEP and FPT_RVM are not explicitly analyzed; the analysis for those requirements is done as part of the activity for the ADV_ARC_EXP component.

11.4 ADV_LLD_(EXP).1

217 The low-level design of a TOE provides a description of the internal workings of the TSF in terms of modules, global data, and their interrelationships. The low-level design is a description of *how* the TSF is implemented to perform its functions, rather than *what* the TSF provides as is specified in the FSP. The low-level design is closely tied to the actual implementation of the TSF, unlike the high-level design, which could be implementation-independent. The primary goal of the low-level design is an aid in understanding the implementation of the TSF, both by reviewing the text of the low-level design as well as a guide when examining the implementation representation (source code).

218 A module is generally a relatively small architectural unit that exhibits properties discussed in ADV_INT_(EXP). A "module" in terms in of the ADV_LLD_EXP requirement refers to the same entity as a "module" for the ADV_INT_EXP requirement.

219 A security-enforcing module is a module that directly implements a security-enforcing TSFI. While this could, for example, include all modules in the call-tree of a security-enforcing module, typically there will be some modules in the call-tree of a security-enforcing module that are not themselves security enforcing. If a module of the TSF is not security enforcing, its implementation still must preserve the security of the TSF; such modules are termed security supporting.

220 A description of a security-enforcing module in the low-level design should be of sufficient detail so that one could create an implementation of the module from the low-level design, and that implementation would

1. be identical to the actual TSF implementation in terms of the interfaces presented and used by the module, and

2. be algorithmically identical to the implementation of the module. For instance, the low-level design may describe a block of processing that is looped over a number of times. The actual implementation may be a *for* loop or a *do* loop, both of which could be used to implement the algorithm. Likewise, a collection of objects could be represented by a linked list or an array; this level of detail is not required to be presented, since both are algorithmically identical. Conversely, if a module's actual implementation performed a bubble sort, it would be inadequate for the low-level design to specify that the module "performed a sort"; it would have to describe the type of sort that was being performed.

221 Security-supporting modules do not need to be described in the same amount of detail, but they should be identified and enough information should be supplied so that 1) the evaluation team can determine that such modules are correctly classified

as security supporting (vs. security enforcing), and 2) the evaluation team has the information necessary to complete the analysis required by ADV_INT_(EXP).1.

222 In the low-level design, security-enforcing modules are described in terms of the interfaces they present to other modules; the interfaces they use (call interfaces) from other modules; global data they access; their purpose; and an algorithmic description of how they provide that function. Security supporting modules are described only in terms of the interfaces they present and their purpose.

223 The interfaces presented by a module are those interfaces used by other modules to invoke the functionality provided. Interfaces are described in terms of how their parameters, and any values that are returned from the interface. In addition to a list of parameters, the descriptions of these parameters are also given. If a parameter were expected to take on a set of values (e.g., a "flag" parameter), the complete set of values the parameter could take on that would have an effect on module processing would be specified. Likewise, parameters representing data structures are described such that each field of the data structure is identified and described. Note that different programming languages may have additional "interfaces" that would be non-obvious; an example would be operator/function overloading in C++. This "implicit interface" in the class description would also be described as part of the low-level design. Note that although a module could present only one interface, it is more common that a module presents a small set of related interfaces.

224 By contrast, interfaces used by a module must be identified such that it can be determined the unique interface that is being invoked by the module being described. It must also be clear from the low-level design the algorithmic reason the invoking module is being called. For instance, if Module A is being described, and it uses Module B's bubble sort routine, an inadequate algorithmic description would be "Module A invokes the double_bubble() interface in Module B to perform a bubble sort." An adequate algorithmic description would be "Module A invokes the double_bubble routine with the list of access control entries; double_bubble() will return the entries sorted first on the username, then on the access_allowed field according the following rules..." The low-level design must provide enough detail so that it is clear what effects Module A is expecting from the bubble sort interface. Note that one method of presenting these called interfaces is via a call tree, and then the algorithmic description can be included in the algorithmic description of the called module.

225 If the implementation makes use of global data, the low-level design must describe the global data, and in the algorithmic descriptions of the modules indicate how the specific global data are used by the module. Global data are identified and described much like parameters of an interface.

226 The purpose a module fulfills is a short description indicating what function the module provides. The level of detail provided should be such that the reader could get a general idea of what the module's function is in the architecture, and to

determine (for security-supporting modules) that it is not a security-enforcing module.

227 As discussed previously, the algorithmic description of the module should describe in an algorithmic fashion the implementation of the module. This can be done in pseudo-code, through flow charts, or informal text. It discusses how the parameters to the interface, global data, and called functions are used to accomplish the result. It notes changes to global data, system state, and return values produced by the module. It is at the level of detail that an implementation could be derived that would be very similar to the actual implementation of the system. It does not need to describe actual implementation artifacts (*do* loops vs. *for* loops, linked lists vs. arrays) if such artifacts are algorithmically identical.

228 It should be noted that source code does not meet the low-level design requirements. Although the low-level design describes the implementation, it *is not* the implementation. Further, the comments surrounding the source code are not sufficient low-level design if delivered interspersed in the source code. The low-level design must stand on its own, and not depend on source code to provide details that must be provided in the low level design (whether intentionally or unintentionally). However, if the comments were extracted by some automated or manual process to produce the low-level design (independent of the source code statements), they could be found to be acceptable if they met all of the appropriate requirements.

229 The ADV_LLD_(EXP).1.2E element in this component defines a requirement that the evaluator determine that the low-level design is an accurate and complete instantiation of the user-visible TOE security functional requirements. This provides a direct correspondence between the TOE security functional requirements and the low-level design, in addition to the pair-wise correspondences required by the ADV_RCR family. Although the evaluator may use the evidence provided in ADV_RCR as an input to making this determination, ADV_RCR cannot be the basis for a positive finding in this area. The requirement for completeness is intended to be relative to the level of abstraction of the low-level design. Note that for this element, FPT_SEP and FPT_RVM are not explicitly analyzed; the analysis for those requirements is done as part of the activity for the ADV_ARC_EXP component.

11.5 ADV_ARC_(EXP).1

230 The architectural design of the TOE is related to the information contained in other decomposition documentation (functional specification, high-level design, low-level design) provided for the TSF, but presents the design in a manner that supports the argument that the TSP cannot be compromised (FPT_SEP) and that it cannot be bypassed (FPT_RVM). The objective of this component is for the developer to provide an architectural design and justification associated with the integrity and non-bypassability properties of the TSF.

231 FPT_SEP and FPT_RVM are distinct from other SFRs because they largely have no directly observable interface at the TSF. Rather, they are properties of the TSF that are achieved through the design of the system, and enforced by the correct implementation of that design. Because of their pervasive nature, the material needed to provide the assurance that these requirements are being achieved is better suited to a presentation separate from the design decomposition of the TSF as embodied in ADV_FSP_EXP, ADV_HLD_EXP, and ADV_LLD_(EXP). This is not to imply that the architectural design called for by this component cannot reference or make use of the design composition material; but it is likely that much of the detail present in the decomposition documentation will not be relevant to the argument being provided for the architectural design document.

232 The architectural design document consists of two types of information. The first is the design information for the entire TSF related to the FPT_SEP and FPT_RVM requirements. This type of information, like the decompositions for ADV_HLD_EXP and ADV_FSP_EXP, describes *how* the TSF is implemented. The description, however, should be focused on providing information sufficient for the reader to determine that the TSF implementation is likely not to be compromised, and that the TSP enforcement mechanisms (that is, those that are implementing SFRs other than FPT_SEP and FPT_RVM) are likely always being invoked.

233 The nature of the FPT_SEP requirement lends itself to a design description much better than FPT_RVM. For FPT_SEP, mechanisms can be identified (e.g., memory management, protected processing modes provided by the hardware, etc.) and described that implement the domain separation. However, FPT_RVM is concerned with interfaces that bypass the enforcement mechanisms. In most cases this is a consequence of the implementation, where if a programmer is writing an interface that accesses or manipulates an object, it is that programmer's responsibility to use interfaces that are part of the TSP enforcement mechanism for the object and not to try to "go around" those interfaces. However, the developer is still able to describe architectural elements (e.g., object managers, macros to be invoked for specific functionality) that pertain to the design of the system to achieve the "always invoked" property of the TSF.

234 For FPT_SEP, the design description should cover how user input is handled by privileged-mode routine; what hardware self-protection mechanisms are used and

how they work (e.g., memory management hardware, including translation lookaside buffers); how software portions of the TSF use the hardware self-protection mechanisms in providing their functions; and any software protection constructs or coding conventions that contribute to meeting FPT_SEP.

235 For FPT_RVM, the description should cover resources that are protected under the SFRs (usually FDP_* components) and functionality (e.g., audit) that is provided by the TSF. The description should also identify the interfaces that are associated with each of the resources or the functionality; this might make use of the information in the FSP. This description should also describe any design constructs, such as object managers, and their method of use. For instance, if routines are to use a standard macro to produce an audit record, this convention is a part of the design that contributes to the non-bypassability of the audit mechanism. It's important to note that "non-bypassability" in this context is not an attempt to answer the question "could a part of the TSF implementation, if malicious, bypass a TSP mechanism", but rather it's to document how the actual implementation does not bypass the mechanisms implementing the TSP.

236 In addition to the descriptive information indicated in the previous paragraphs, the second type of information an architectural design document must contain is a justification that the FPT_SEP and FPT_RVM requirements are being met. This is distinct from the description, and presents an argument for why the design presented in the description is sufficient.

237 For FPT_SEP, the justification should cover the possible modes by which the TSF could be compromised, and how the mechanisms implemented in response to FPT_SEP counter such compromises. The vulnerability analysis might be referenced in this section.

238 For FPT_RVM, the justification demonstrates that whenever a resource protected by an SFR is accessed, the protection mechanisms of the TSF are invoked (that is, there are no "backdoor" methods of accessing resources that are not identified and analyzed as part of the ADV_FSP_EXP/ADV_HLD_EXP/ADV_LLD_EXP analysis). Similarly, the description demonstrates that a function described by an SFR is always provided where required. For example, if the FCO_NRO family were being used the description should demonstrate that all interfaces either 1) do not deal with transmitting the information identified in the FCO_NRO component included in the ST, or 2) invoke the mechanism(s) described by the decomposition documentation. The justification for FPT_RVM will likely need to address all of the TSFI in order to make the case that the TSP is non-bypassable.

12 REFINEMENTS

239 This section contains refinements where text was omitted. Omitted text is shown as bold text within parenthesis. The actual text of the functional requirements as presented in Section 5 has been retained.

FCS_CKM.1.1 (1) **Refinement:** The TSF shall generate **symmetric** cryptographic keys in accordance with a specified cryptographic key generation algorithm as follows: [selection:

- A hardware random number generator (RNG) as specified in FCS_COP_(EXP).1, but with a NIST-approved hashing function required for mixing, and/or

- A software RNG as specified in FCS_COP_(EXP).1,

(and specified cryptographic key sizes [*assignment: cryptographic key sizes*])

That meets the following:

- FIPS PUB 180-2, Secure Hash Algorithm

FCS_CKM.1.1 (2) **Refinement:** The TSF shall generate **asymmetric** cryptographic keys in accordance with a **(specified key generation algorithm) domain parameter generator** and ***[selection:***

- a random number generator and/or

- a prime number generator].

(and specified cryptographic key sizes [assignment*: key sizes*])that meet the following:

- Generated key strength shall be equivalent to, or greater than, a symmetric key strength of 128 bits using conservative estimates;

- ANSI X9.80 (3 January 2000), Prime Number Generation, Primality Testing, and Primality Certificates using random integers with deterministic tests, or constructive generation methods;

- Case: For domain parameters used in finite field-based key establishment schemes

    - ANSI X9.42-2001, Public Key Cryptography for the Financial Services Industry: Agreement of Symmetric Keys Using Discrete Logarithm Cryptography;

- Case: For domain parameters used in RSA-based key establishment schemes (with odd e)
  - ANSI X9.31-1998 (May 1998), Digital Signatures Using Reversible Public Key Cryptography for the Financial Services Industry (rDSA) for the generation of the RSA parameters; and
- Case: For domain parameters used in elliptic curve-based key establishment schemes
  - ANSI X9.63-200x (1 Oct 2000), Public Key Cryptography for the Financial Services Industry: Key Agreement and Key Transport using Elliptic Curve Cryptography

FCS_CKM.4.1 **Refinement:** The TSF shall destroy cryptographic keys in accordance with a **(specified)** cryptographic key **(destruction)** zeroization method **([assignment: *cryptographic key destruction method*])** that meets the following:

- FIPS PUB 140-2;
- Zeroization of all plaintext cryptographic keys and all other critical cryptographic security parameters shall be immediate and complete; and
- For embedded cryptographic modules, the zeroization shall be executed by overwriting the key/critical cryptographic security parameter storage area three or more times using a different alternating data pattern each time.

FCS_COP.1.1(1) **Refinement**: The TSF shall perform **data encryption/decryption services** in accordance with a **(specified cryptographic) NIST-approved implementation of** the cryptographic algorithm **Triple Data Encryption Algorithm (TDEA) used in NIST-approved modes of operation** and cryptographic key size **of 168 bits (three independent keys)** that meets the following:

- FIPS PUB 140-2, security Requirements for Cryptographic Modules,
- FIPS PUB 46-3, Data Encryption Standard, and
- ANSI X9.52-1998, Triple Data Encryption Algorithm Modes of Operation.

FCS_COP.1.1(2) **Refinement:** The TSF shall perform **cryptographic signature services** in accordance with the **(specified cryptographic) NIST-approved digital signature algorithm** [selection:

- Digital Signature Algorithm (DSA) with a key size (modulus) of 2048 bits or greater,

- RSA Digital Signature Algorithm (rDSA with odd e) with a key size (modulus) of 2048 bits or greater, or
- Elliptic Curve Digital Signature Algorithm (ECDSA) with a key size of 256 bits or greater].

(and cryptographic key sizes) that meet the following:

- Case: Digital Signature Algorithm

  FIPS PUB 186-2, Digital Signature Standard, for signature creation and verification processing; and ANSI Standard X9.42-2001, Public Key Cryptography for the Financial Services Industry: Agreement of Symmetric Keys Using Discrete Logarithm Cryptography for generation of the domain parameters;

- Case: RSA Digital Signature Algorithm (with odd e)

  .ANSI X 9.31-1998 (May 1998), Digital Signatures Using Reversible Public Key Cryptography For The Financial Services Industry (rDSA);

- Case: Elliptic Curve Digital Signature Algorithm

  ANSI X9.62-1-xxxx (10 Oct 1999), Public Key Cryptography for the Financial Services Industry: Elliptic Curve Digital Signature Algorithm (ECDSA).

FCS_COP.1.1(3) **Refinement:** The TSF shall perform cryptographic hashing services in accordance with a **(specified cryptographic)** NIST-approved hash implementation of the Secure Hash algorithm and message digest size of at least 256 bits that meets the followings: FIPS PUB 180-2.

FCS_COP.1.1(4) **Refinement:** The TSF shall perform **cryptographic key agreement services** in accordance with a **(specified cryptographic) NIST-approved implementation of a key agreement** algorithm *[selection:*

- Finite Field-based key agreement algorithm and cryptographic key sizes(modulus) of 2048 bits or greater,
- Elliptic Curve-based key agreement algorithm and cryptographic key size of 256 bits or greater]

(and cryptographic key sizes) that meets the following:

- Case: Finite field-based key agreement schemes

ANSI X9.42-2001, Public Key Cryptography for the Financial Services Industry: Agreement of Symmetric Keys Using Discrete Logarithm Cryptography:

- Case: Elliptic curve-based key agreement schemes

  ANSI X9.63-200x (1 Oct 2000), Public Key Cryptography for the Financial Services Industry: Key Agreement and Key Transport using Elliptic Curve Cryptography.

13 STATISTICAL RANDOM NUMBER GENERATOR TESTS

A cryptographic module employing random number generators (RNGs) shall perform the following statistical tests for randomness. A single bit stream of 20,000 consecutive bits of output from each RNG shall be subjected to the following four tests: monobit test, poker test, runs test, and long runs test. (These four tests are simply those that formerly existed as the statistical RNG tests in Federal Information Processing Standard 140-2. However, for purposes of meeting this protection profile, these tests must be performed at the frequency specified earlier in this protection profile.)

The Monobit Test:

1. Count the number of ones in the 20,000 bit stream. Denote this quantity by X.
2. The test is passed if 9,725 < X < 10,275.

The Poker Test:

1. Divide the 20,000 bit stream into 5,000 contiguous 4 bit segments. Count and store the number of occurrences of the 16 possible 4 bit values. Denote *f(i)* as the number of each 4 bit value *i*, where 0 < *i* < 15.
2. Evaluate the following:

$$X = (16 / 5000) * \left(\sum_{i=0} [f(i)]^2 \right) - 5000$$

3. The test is passed if 2.16 < X < 46.17.

The Runs Test:

1. A run is defined as a maximal sequence of consecutive bits of either all ones or all zeros that is part of the 20,000 bit sample stream. The incidences of runs (for both consecutive zeros and consecutive ones) of all lengths (> 1) in the sample stream should be counted and stored.
2. The test is passed if the runs that occur (of lengths 1 through 6) are each within the corresponding interval specified in the table below. This must hold for both the zeros and ones (i.e., all 12 counts must lie in the specified interval). For the purposes of this test, runs of greater than 6 are considered to be of length 6.

Table C.1 - Required Intervals for Length of Runs Test

Length of Run	Required Interval
1	2343 - 2657
2	1135 - 1365
3	542 - 708
4	251 - 373

5	111 - 201
6 and greater	111 - 201

The Long Runs Test:

1. A long run is defined to be a run of length 26 or more (of either zeros or ones).
2. On the sample of 20,000 bits, the test is passed if there are no long runs.

14 RANDOMIZER QUALIFICATION TESTING REQUIREMENTS

This test utilizes the NIST battery of statistical tests as described in "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications", NIST Special Publication 800-22. This document and corresponding software code are available for downloading at the following Internet sites: http://csrc.ncsl.nist.gov/rng or http://csrc.ncsl.nist.gov/CryptoToolkit/tkrng .
297 The Randomizer Qualification Statistical Test Suite consists of the following statistical tests:

1. Frequency (Monobit) Test
2. Frequency Test within a Block
3. Cumulative Suns (Cusum) Test
4. Runs Test
5. Longest Run of ones in a Block
6. Binary Matrix Rank Test
7. Discrete Fourier Transform (Spectral) Test
8. Maurer's Universal Statistical Test
9. Approximate Entropy Test
10. Serial Test

Randomizer Qualification Test Process

Power up the randomizer and collect a sample of 100,000 bits of data every 5 minutes until 10 samples have been collected. Concatenate the 10 samples to form a single sample of length 1,000,000 bits. Apply the above statistical tests using the following input parameters:

Sequence Length: 100,000
Number of Sequences: 10
Block Frequency Test Block Length: 100
Universal Test Block Length: 6
Universal Test Number of Initialization Steps: 640
Approximate Entropy Block Length: 10
Serial Test Block Length: 10

Each statistical test will produce a series of 10 P-Values. The Cusum and Serial test consist of two tests each and produces two series of 10 P-Values each. Thus the statistical test suite will produce twelve series of 10 P-Values each. The collected sample of data passes the statistical test suite if for each of the twelve series of P-Values at least 9 of the 10 P-Values are greater than 0.01. The NIST software generates a file, FinalAnalysisReport, which summarizes the results of the tests. The data passes the statistical test suite if all of the twelve values listed in the proportions column are greater than or equal to 0.9.

The above test procedure is to be repeated 3 times. The randomizer passes the randomizer qualification test if the statistical test suite is passes on at least 2 of the 3 attempts.